

Méthodes algébriques pour la formalisation et l'analyse de politiques de sécurité

Tony Bourdier

▶ **To cite this version:**

Tony Bourdier. Méthodes algébriques pour la formalisation et l'analyse de politiques de sécurité. Logique en informatique [cs.LO]. Université Henri Poincaré - Nancy I, 2011. Français. NNT : . tel-00646401

HAL Id: tel-00646401

https://theses.hal.science/tel-00646401

Submitted on 16 Dec 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

École doctorale Informatique, Automatique, Électronique et Mathématiques — DFD Informatique
Université Henri Poincaré – Nancy 1 — Faculté des Sciences et Technologies

Méthodes algébriques pour la formalisation et l'analyse de politiques de sécurité

THÈSE

soutenue le 7 octobre 2011 pour l'obtention du titre de

Docteur de l'Université Henri Poincaré – Nancy 1

(spécialité informatique)

par

Tony Bourdier

Composition du jury

Rapporteurs :	M	David Delahaye	MdC HDR au CNAM de Paris
	Mme	Marie-Laure Potet	PR à l'INP de Grenoble
Examinateurs :	Mme	Isabelle Chrisment	PR à l'Université de Nancy 1
	Mme	Maribel Fernandez	PR au King's College de Londres
	Mme	Hélène Kirchner	DR à l'INRIA Paris - Rocquencourt
	Mme	Olga Kouchnarenko	PR à l'Université de Franche-Comté
Directeur de thèse :	M	Horatiu Cirstea	PR à l'Université de Nancy 1
Invité :	M	Mathieu Jaume	MdC HDR à l'Université de Paris 6

Laboratoire Lorrain de Recherche en Informatique et ses Applications

À ma fille Charlie

Table des matières

1 Introduction et positionnement **5**

1.1 Les mécanismes de sécurité . . . 7
1.1.1 La cryptologie . . . 7
1.1.2 L'authentification . . . 7
1.1.3 La gestion des autorisations : les politiques de sécurité . . . 8
1.2 Un besoin de formalisation . . . 10
1.2.1 Exemple introductif . . . 10
1.2.2 Évaluation de la sécurité . . . 11
1.3 Les modèles de sécurité, petit historique . . . 13
1.3.1 1971 – La matrice de droits d'accès de Lampson . . . 13
1.3.2 1973 – Le modèle de Bell et LaPadula . . . 14
1.3.3 1976 – Le modèle de Harrison, Ruzzo et Ullman . . . 15
1.3.4 1996 – Role Based Access Control . . . 15
1.4 Positionnement . . . 16
1.5 Feuille de route et contributions . . . 17

2 Fondements techniques **21**

2.1 Notations et définitions de base . . . 22
2.1.1 Ensembles . . . 22
2.1.2 Relations . . . 22
2.1.3 Morphismes . . . 24
2.1.4 Relations binaires . . . 24
2.1.5 Induction . . . 26
2.2 Algèbre des termes . . . 27
2.2.1 Termes . . . 27
2.2.2 Σ-algèbre . . . 32

2.2.3 Spécification équationnelle . . . 33
2.2.4 Problème du mot, raisonnement équationnel . . . 35
2.3 Réécriture . . . 36
2.3.1 Définitions . . . 37
2.3.2 Preuves de propriétés . . . 38
2.4 Automates finis d'arbres . . . 39
2.4.1 Langages d'arbres . . . 39
2.4.2 Langages de tuples . . . 42
2.5 Logique du premier ordre . . . 45
2.5.1 Syntaxe . . . 45
2.5.2 Sémantique . . . 47
2.5.3 Sémantiques syntaxiques et constructeurs . . . 48

3 Spécification, analyse et transformation de politiques de sécurité 49
3.1 Spécification des modèles et des politiques . . . 50
3.1.1 Vocabulaire . . . 50
3.1.2 Modèles de sécurité . . . 51
3.1.3 Politiques de sécurité . . . 54
3.2 Sémantique et propriétés . . . 57
3.3 Changement de modèle . . . 60
3.3.1 Positionnement du problème . . . 60
3.3.2 Méthode . . . 62
3.4 Travaux reliés . . . 67
3.5 Synthèse . . . 69

4 Spécification et vérification de politiques dynamiques 71
4.1 Systèmes sécurisés et politiques dynamiques . . . 72
4.1.1 Théorie de sécurité . . . 73
4.1.2 Environnements et règles de transition . . . 76
4.1.3 Système de sécurité . . . 81
4.1.4 Politiques de sécurité . . . 83
4.1.5 Système sécurisé . . . 85
4.2 Sémantiques basées sur la réécriture . . . 85
4.2.1 Représentation symbolique des environnements . . . 86
4.2.2 Évaluation des contraintes . . . 89
4.2.3 Systèmes de sécurité . . . 93
4.2.4 Politiques de sécurité . . . 95
4.2.5 Système sécurisé . . . 96
4.3 Vérification . . . 97
4.3.1 Propriétés de politiques . . . 97
4.3.2 Le problème de l'administration . . . 98
4.3.3 Comparaison de politique . . . 99
4.3.4 Analyse par requêtage . . . 100
4.4 Travaux reliés . . . 102
4.5 Synthèse . . . 103

5 Les politiques de sécurité dans les réseaux informatiques 105
5.1 Firewalls : une introduction informelle . . . 107
5.2 Spécification formelle de firewalls . . . 111
5.2.1 Vocabulaire et définition . . . 111
5.2.2 Sémantique basée sur les automates d'arbres . . . 113
5.3 Applications . . . 118
5.3.1 Propriétés . . . 119
5.3.2 Analyse structurelle . . . 120
5.3.3 Analyse par requêtage . . . 125
5.4 Composition de firewalls . . . 126
5.4.1 Topologies de réseaux . . . 126
5.4.2 Politiques de sécurité réseau . . . 128
5.4.3 Stratégie de routage . . . 130
5.4.4 Sémantique par réécriture . . . 131
5.4.5 Propriétés et analyse . . . 140
5.5 Travaux reliés . . . 145
5.6 Synthèse . . . 146

6 Synthèse 147
6.1 Contributions . . . 147
6.2 Perspectives . . . 149

1 Introduction et positionnement

Sommaire

1.1 Les mécanismes de sécurité 7
1.1.1 La cryptologie 7
1.1.2 L'authentification 7
1.1.3 La gestion des autorisations : les politiques de sécurité . . . 8
1.2 Un besoin de formalisation 10
1.2.1 Exemple introductif 10
1.2.2 Évaluation de la sécurité 11
1.3 Les modèles de sécurité, petit historique 13
1.3.1 1971 – La matrice de droits d'accès de Lampson 13
1.3.2 1973 – Le modèle de Bell et LaPadula 14
1.3.3 1976 – Le modèle de Harrison, Ruzzo et Ullman 15
1.3.4 1996 – Role Based Access Control 15
1.4 Positionnement 16
1.5 Feuille de route et contributions 17

L'information représente de nos jours l'un des facteurs essentiels de la performance des entreprises et des économies nationales et internationales [Thépaut, 2002]. L'entrepreneur comme le politique cherche à mettre la main sur toutes les informations qui peuvent lui être utiles, et ce, en employant des moyens variés, allant des opérations les plus honnêtes à l'espionnage industriel voire diplomatique [Russo, 1966]. Ce pouvoir informationnel et les bénéfices qui en découlent prennent

un relief particulier à l'heure où le monde procède à une véritable mutation numérique. Hier, la presse traditionnelle était qualifiée de quatrième pouvoir. Aujourd'hui, les experts de l'informatique sont en passe de devenir les nouveaux maîtres de l'information et par là même les nouveaux arbitres de la conquête du pouvoir économique et politique. Par ailleurs, cette révolution informatique crée les conditions favorables à l'accroissement de la transmission des informations, certifiées ou non, faisant s'échapper tout contrôle sur les données dès lors que ces dernières parviennent entre les mains d'un tiers non digne de confiance. Dans ce contexte, la problématique de gestion de l'information devient celle de la sécurité des systèmes d'information.

La nécessité de sécurisation des systèmes d'information a amené la commission européenne à établir un standard concernant les besoins de sécurité correspondants [Commission of the European Communities, 1991]. C'est ainsi qu'ont été définis trois principaux objectifs servant aujourd'hui de référentiel pour la communauté de la sécurité informatique. Le premier objectif porte le nom de confidentialité et se définit comme l'aptitude d'un système à prévenir toute divulgation non autorisée d'une information. Le système (informatique) doit pouvoir empêcher les utilisateurs non autorisés à lire une information confidentielle. Le second objectif, appelé intégrité, consiste à empêcher l'altération non autorisée d'une information. Enfin, le troisième objectif, la disponibilité, concerne l'assurance pour les utilisateurs de pouvoir accéder aux informations qui leur sont autorisées ; il s'agit donc de prévenir du risque de déni de service. Il est à remarquer que les définitions de ces trois propriétés s'entendent relativement à la notion d'information. Car, contrairement à ce que l'on pourrait penser en première instance, la notion d'information ne réfère pas nécessairement au contenu d'un fichier informatique. Les connaissances relatives aux accès aux fichiers, leur création, etc. ... constituent également des informations. Dans ce contexte, la notion d'information s'impose comme un référentiel pour comprendre et appréhender la sémantique des trois objectifs sus-cités. D'autres propriétés, comme l'anonymat (non divulgation de l'identité) ou la non répudiation (impossibilité pour l'émetteur et le destinataire d'une information de nier avoir respectivement envoyé et reçu ladite information), s'entendent ainsi respectivement comme des instances des notions de confidentialité et d'intégrité relativement à des notions d'information qui ne concernent pas le contenu d'un fichier informatique (dans ce cas, la « paternité » d'une action constitue l'information d'intérêt). Par ailleurs, selon les domaines d'application, l'accent est mis sur des propriétés différentes. Les considérations militaires sont focalisées sur les problématiques de confidentialité tandis que les enjeux financiers nécessitent des exigences de sécurité en terme d'intégrité.

Le présent chapitre a pour mission d'introduire les différentes notions relatives à la sécurité informatique. Nous y dressons un portrait général et relativement succint des problématiques sous-jacentes. Pour une description plus détaillée des notions abordées dans ce chapitre, le lecteur est invité à consulter les références suivantes qui ont très largement inspiré l'écriture de ce chapitre : [Bishop, 2002, Mé and Deswarte, 2006, Deswarte and Mé, 2002, Abrams et al., 1995, ANSSI, 2011, Dausque, 2005].

1.1 Les mécanismes de sécurité

Pour garantir les propriétés de confidentialité, d'intégrité et de disponibilité, trois principaux mécanismes ont été développés : la cryptologie, les procédures d'authentification et la gestion des autorisations.

1.1.1 La cryptologie

La cryptologie, étymologiquement la science du secret, se compose des deux disciplines que sont la cryptographie, l'art d'écrire des secrets pour les rendre inintelligibles à des tiers, et la cryptanalyse, l'art de retrouver des secrets cachés dans des informations inintelligibles. Selon l'objectif de sécurité ciblé (confidentialité ou intégrité), les méthodes utilisées sont différentes. Lorsqu'il s'agit d'assurer la confidentialité d'un message, on a recours aux méthodes dites de chiffrement. La méthode de chiffrement la plus célèbre est sans doute celle qu'utilisait Jules César pour ses correspondances secrètes. L'algorithme correspondant, rudimentaire, est parfois appelé chiffre (ou substitution) de César et consiste à substituer dans un texte chaque lettre par la $n^{\text{ème}}$ lettre suivante où n, la clé, est un entier fixé (étant entendu que A suit la lettre Z). Par exemple, en définissant la clé à 3, la phrase « Cras ero Scientiæ Doctor [1] », est chiffrée en « Fudv hur Vflhqwldh Grfwru ». La nature simpliste de l'algorithme nécessitait de tenir secret la technique de chiffrement pour conserver un intérêt. D'autres algorithmes beaucoup plus évolués permettent de garantir la confidentialité d'un message même si la méthode de chiffrement est publique. C'est le cas des méthodes de chiffrement dites asymétriques, nécessitant que chaque utilisateur définisse une clé publique, destinée au chiffrement, ainsi qu'une clé privée associée, destinée au déchiffrement. La clé publique doit être utilisée par quiconque souhaitant envoyer un message confidentiel à l'utilisateur qui pourra alors déchiffrer le message avec la clé privée dont il est le seul à disposer. D'autres techniques, comme les fonctions de hachage ou les signatures, permettent de garantir l'intégrité d'un message. Ces techniques ne permettent pas de reconstruire le message d'origine mais de fournir une preuve que le message d'origine n'a pas été altéré. Elles consistent à associer à chaque message une empreinte de taille fixe telle que :

- l'empreinte soit facilement calculable à partir du message d'origine,
- il ne soit pas possible de déduire le message à partir de l'empreinte et
- la probabilité que deux messages différents possèdent la même empreinte soit quasi-nulle.

Les fonctions de hachage permettent par exemple l'usage de mécanismes d'authentification par mot de passe sans qu'il soit nécessaire pour le système de conserver ce dernier (*c.f.* section suivante).

1.1.2 L'authentification

Tout utilisateur d'un système informatique agit sous une certaine identité. L'authentification consiste à vérifier l'identité dont une entité (personne ou machine) se réclame, c'est-à-dire l'association entre une entité physique et son identité virtuelle.

1. Littéralement « demain je serai docteur ès sciences ».

Pour pouvoir procéder à une authentification, il est nécessaire d'avoir au préalable fourni au système un moyen de se faire reconnaître. Ce moyen consiste, en général, à vérifier que l'utilisateur possède une preuve de son identité sous l'une des formes suivantes :

- lui poser une question dont il est le seul à connaître la réponse (mot de passe, numéro d'identification personnel, ...)
- lui demander de fournir un élément qu'il est le seul à posséder (carte d'identité, certificat électronique, badge, ...)
- analyser des caractéristiques de l'utilisateur qu'il doit être le seul à posséder (caractéristiques physiques, biométrie, ...)
- vérifier sa localisation (adresse IP, GPS, ...)

La procédure d'authentification consiste donc pour le système à effectuer l'une de ces vérifications, ce qui signifie qu'il doit posséder certaines informations concernant l'utilisateur. Formellement, un processus d'authentification doit être composé des éléments suivants [Bishop, 1991] :

- un ensemble A d'informations d'authentification spécifiques à partir desquelles les utilisateurs doivent prouver leur identité,
- un ensemble C d'informations complémentaires dont le système dispose et qui lui servent à établir la validité de la preuve fournie par l'utilisateur,
- une fonction $f : A \to C$ dite de complémentation qui génère l'information complémentaire à partir de l'information d'authentification,
- un ensemble V de fonctions $v : A \times C \to \{0, 1\}$ vérifiant l'adéquation entre une information d'authentification et une information complémentaire,
- un ensemble de fonctions S dites de sélection permettant à un utilisateur de créer ou de modifier les informations d'authentification et complémentaires.

Dans les précédentes versions d'UNIX, les utilisateurs s'identifiaient avec un mot de passe comportant au maximum 8 caractères. Le système stockait (dans le fichier `/etc/passwd`) l'image des mots de passe par une fonction de hachage. Ainsi, en reprenant les notations précédemment introduites, A comprend l'ensemble des chaînes de charactères de longueur inférieure ou égale à 8, C comporte l'ensemble des chaînes de charactères de longueur 13, f est une fonction de hachage, V correspond à la fonction `login` et S à la fonction `passwd`.

Le processus d'authentification est souvent une étape préalable aux autres mécanismes de sécurité, en particulier la gestion des autorisations suppose qu'un tel processus est en place.

1.1.3 La gestion des autorisations : les politiques de sécurité

Pour garantir une propriété de sécurité, il est nécessaire de ne pas laisser les utilisateurs libres d'effectuer sur le système n'importe quelle action. En fonction de leur statut ou du degré de confiance qui leur est accordé, les possibilités d'actions qu'un utilisateur peut entreprendre doivent être différentes. Dans ce cadre, il est nécessaire de mettre en place un processus de gestion des autorisations d'actions, aussi appelé politique de sécurité. Il est à noter que pour qu'un tel processus ait un intérêt, le système doit être doté d'un mécanisme d'authentification garantissant l'identité des individus. Une politique de sécurité doit ainsi exprimer l'ensemble des

règles régissant les actions de chacun afin que le système satisfasse les propriétés de sécurité voulues. Par ailleurs, les définitions de la notion de politique de sécurité dans la littérature sont très diverses et se situent à des niveaux d'abstraction et de raisonnement parfois différents. Selon la référence choisie, une politique de sécurité se réfère à :

- un ensemble de règles, lois et pratiques régissant le comportement des utilisateurs,
- un document de référence permettant de décrire les mesures élémentaires de protection et d'utilisation du système d'information afin que tout le monde puisse les respecter,
- un document de référence définissant les objectifs à atteindre et les moyens accordés pour y parvenir,
- aux propriétés que doit vérifier le système pour être considéré comme sûr.

La diversité des définitions données du terme politique de sécurité nous fait penser qu'il s'agit plus d'un concept que d'une entité précise. Dans son article « On the Buzzword "Security Policy" » [Sterne, 1991], Sterne établit que toute définition d'une politique de sécurité doit pouvoir identifier les éléments fondamentaux suivants :

1. l'objectif de sécurité,
2. la politique de sécurité organisationnelle et
3. la politique de sécurité automatisée.

qu'il définit de la façon suivante :

Définition (Objectif de sécurité). Un objectif de sécurité est une expression de l'intention de protéger une ressource identifiée d'une utilisation non autorisée. Cette expression doit identifier le type d'actions à réguler. La ressource identifiée doit être concrète ou avoir une forme concrète. Un objectif de sécurité n'a de sens pour une organisation seulement si l'organisation en question possède ou contrôle la ressource à protéger.

Cette définition a pour but d'établir la notion de sécurité à partir de laquelle les autres définitions vont se positionner. La protection d'informations classées contre la divulgation à des tiers non autorisés est un exemple d'objectif de sécurité, au même titre que la prévention contre la vente illicite d'armes ou d'actifs financiers. C'est donc sur la base de la définition d'un objectif de sécurité que repose celle de politique de sécurité organisationnelle.

Définition (Politique de sécurité organisationnelle). Une politique de sécurité organisationnelle est constituée d'un ensemble de lois, de règles et de pratiques qui régulent la manière dont une organisation gère, protège et distribue les ressources pour atteindre les objectifs de sécurité. Ces lois, règles et pratiques doivent identifier des critères d'attribution des droits et spécifier les conditions sous lesquelles chaque individu est autorisé à exercer ou déléguer ses droits. Pour avoir un sens, ces lois, règles et pratiques doivent permettre aux individus de déterminer raisonnablement si leurs actions violent ou sont conformes à la politique.

Par opposition à l'objectif de sécurité qui constitue une déclaration d'intention relativement abstraite, la politique organisationnelle décrit la façon dont l'objectif de sécurité se manifeste dans les activités quotidiennes de l'organisation. Enfin, Sterne explique que la définition d'une telle politique doit se décliner en une politique de sécurité dite automatisée.

Définition (Politique de sécurité automatisée). Une politique de sécurité automatisée se définit par un ensemble de restrictions et de propriétés spécifiant la manière dont un système informatique se prémunit d'une utilisation des informations et des ressources informatiques dédiée à la violation de la politique de sécurité organisationnelle. Cet ensemble doit être accompagné d'arguments d'ingénierie persuasifs démontrant que ces restrictions jouent un rôle clé dans la mise en oeuvre de la politique organisationnelle.

Cette dernière définition est le pendant opérationnel de la politique organisationnelle dans le cadre d'une gestion informatisée de l'information.

1.2 Un besoin de formalisation

Admettons dans notre cas que la notion de politique de sécurité fait référence à un ensemble de règles permettant de déterminer si un individu est autorisé à effectuer une action donnée. Ces règlements sont généralement décrits dans un langage naturel relativement simple (pour être compréhensible par tous) et sont la plupart du temps sujets à ambiguïté. Il arrive en effet que certaines notions apparaissant dans un tel règlement soient floues voire non définies.

1.2.1 Exemple introductif

Prenons l'exemple suivant décrivant la politique de sécurité régissant l'usage de la messagerie électronique à l'université de l'état de l'Illinois Nord.

> *Exemple.* [Northern Illinois University, 2010] L'usage de listes de diffusion est autorisé sous les conditions suivantes :
>
> 1. toute liste de diffusion comprenant plus de 100 destinataires doit avoir :
>   - l'aval du doyen (ou d'un représentant) si l'ensemble des destinataires comprend des étudiants qui n'appartiennent pas au groupe des administrateurs de la liste de diffusion,
>   - l'aval du vice-président de l'université ou du service des ressources humaines si l'ensemble des destinataires comprend des employés qui n'appartiennent pas au groupe des administrateurs de la liste de diffusion,
> 2. le contenu du message doit être inclus dans le message lui-même et pas en pièce jointe dans la mesure du possible,
> 3. le message doit préférentiellement contenir des liens plutôt que des pièces jointes afin d'éviter d'augmenter la taille du message.

> L'usage des emails est interdit dans les contextes suivants (liste non exhaustive) :
>
> 4. l'objectif du message envoyé viole une loi fédérale,
> 5. le message est à caractère commercial,
> 6. l'identité de l'émetteur n'apparaît pas,
> 7. l'envoi de masse congestionnant le réseau,
> 8. la diffusion de messages inappropriés à des listes ou des individus,
> 9. attribution d'une priorité « élevée » à un message envoyé sur une liste de diffusion,
> 10. ...
>
> Si certaines règles sont sans ambiguïté (règles 1 et 6), certaines sont sujettes à interprétation (règles 3 et 8). D'autres peuvent également paraître en conflit. Par exemple, si le doyen souhaite envoyer un message étiqueté comme urgent (pour éviter qu'il ne soit directement jeté à la corbeille sans être lu par une partie des destinataires) à l'ensemble de l'université pour les informer de la fermeture temporaire de l'établissement par mesure de sécurité. La première règle le lui permet alors que la neuvième le lui interdit. Sans compter que la politique précise que la liste des interdictions n'est pas exhaustive.

Vous l'aurez compris, l'établissement de politiques (ou règlements) de sécurité nécessite un cadre plus formel que le simple usage du langage naturel pour permettre à chacun de déterminer sans ambiguïté si une action lui est permise ou non.

1.2.2 Évaluation de la sécurité

Dans un contexte de généralisation de l'usage des technologies informatiques pour le traitement et la manipulation de l'information, il devient nécessaire d'être en mesure d'assurer qu'un système informatique donné satisfait réellement les exigences de sécurité établies. Il semble évident que l'utilisateur d'un système informatique a besoin d'avoir confiance en la sécurité du système qu'il utilise. Effectueriez-vous des opérations bancaires en ligne si vous n'aviez pas confiance dans le système de protection que votre banque vous assure avoir mis en place ? Rempliriez-vous votre déclaration de revenus sur internet si vous n'étiez pas intimement persuadé que personne hormis les fonctionnaires habilités n'aura accès aux données entrées ? Sans le savoir, nous procédons intellectuellement à une évaluation du niveau de sécurité des systèmes qui nous sont proposés. Dans le cas d'un usage personnel d'un système, cette évaluation sera très majoritairement basée sur la notoriété du « constructeur » du système. Mais dans un cadre plus critique, comme par exemple le système d'information du ministère de la défense ou de celui de l'économie et des finances, il est nécessaire de disposer d'un système d'évaluation basé sur des considérations objectives. Un tel système d'évaluation a été mis en place dans ce que l'on nomme les critères communs [ISO, 2009]. Ce système définit une échelle comportant sept niveaux d'assurance connue sous le nom d'EAL (Evaluation Assurance Level). Ces niveaux permettent d'acquérir une certaine confiance dans le produit relativement

Critères communs

Common Criteria (ou Critères Communs en français) est un standard international (ISO/CEI 15408) pour la sécurité des systèmes d'information. Le nom complet du standard est Common Criteria for Information Technology Security Evaluation.

au besoin de sécurité défini dans la « cible de sécurité ». Une augmentation du niveau EAL nous assure une augmentation des exigences de sécurité, des vérifications faites par l'évaluateur ainsi que du niveau considéré des éventuels attaquants. Actuellement, les critères communs définissent cinq niveaux de résistance : la vulnérabilité publique, l'attaque élémentaire, l'attaque élémentaire renforcée, l'attaque moyenne et l'attaque élevée.

EAL1 : testé fonctionnellement. Le niveau EAL1 peut être attribué à un système lorsque une certaine confiance est nécessaire dans le bon fonctionnement de ce dernier mais que les menaces ne sont pas considérées comme importantes. Il présente un intérêt lorsqu'une assurance établie de façon indépendante est nécessaire pour démontrer que l'attention requise a été exercée à l'égard de la protection des informations personnelles ou assimilées. Ce niveau a pour but de tester un programme exécutable sur des cas limites et anormaux que l'utilisateur doit lui-même définir.

EAL2 : testé structurellement. L'attribution du niveau EAL2 signifie que la solution informatique a passé avec succès les tests au niveau de ses fonctionnalités et de sa structure, permettant de voir la manière dont sont organisées les différentes couches de sécurité. Le test de fonctionnement a pour objectif de vérifier que le programme se comporte correctement dans son ensemble et plus particulièrement dans un cas normal d'utilisation. Pour effectuer ce test, des entrées sont générées de façon aléatoire et sont soumises au programme à tester.

EAL3 : testé et vérifié méthodiquement. Ce niveau permet d'assurer une fiabilité maximale depuis le stade de la conception avec peu de modifications dans le processus de test. Cette fiabilité maximale implique que le produit ait été conçu dès le départ dans l'optique de répondre à des exigences de sécurité, à la différence d'une mise en oeuvre postérieure à la conception du produit. Le développeur doit prouver qu'il a effectué les tests requis en fournissant des analyses de vulnérabilité. Il s'agit du premier niveau à comporter des critères d'évaluation de la gestion de la configuration.

EAL4 : conçu, testé et vérifié méthodiquement. Ce niveau requiert une visibilité complète du produit. Il impose l'établissement de spécifications complètes des fonctions de sécurité, de leur conception rigoureuse et d'un audit d'un échantillon du code source. L'analyse doit être confortée par des éléments de preuve des tests du développeur basés sur des spécifications fonctionnelles de la conception de haut niveau et par une analyse de vulnérabilité démontrant la résistance à des attaques élémentaires renforcées.

EAL5 : spécifié de façon semi-formelle et testé. À ce niveau, le développeur doit pouvoir expliquer les spécifications de la conception et la manière dont celles-ci sont mises en oeuvre au niveau fonctionnel. Il requiert le développement de spécifications semi-formelles complètes et une conception préliminaire semi-formelle. Des

tests unitaires, d'intégration et de validation doivent avoir été effectués et le produit devra avoir démontré sa résistance aux attaques moyennes.

EAL6 : conçu de façon semi-formelle et testé. Pour atteindre ce niveau de certification, en plus des exigences du niveau EAL5, les politiques de sécurité implantées doivent avoir été spécifiées formellement et le produit doit démontrer sa résistance aux attaques de niveau élevé. Une modularité complète dans la conception est exigée à partir de ce niveau.

EAL7 : formellement spécifié et conception vérifiée formellement. Le niveau EAL7 exige que le produit ait été formellement spécifié et qu'une conception préliminaire formelle ait été effectuée. Ce niveau garantit en particulier que les applications sont parfaitement isolées les unes des autres et que ce processus a été démontré mathématiquement.

Pour atteindre de hauts niveaux de certification (EAL5, 6 et 7), il est donc nécessaire de fournir une spécification formelle de la politique de sécurité et des mécanismes utilisés pour la mettre en oeuvre. C'est ce contexte d'exigence croissante pour la fiabilité et la sûreté des applications de sécurité qui a favorisé l'émergence d'un nouveau domaine de recherche : celui de la modélisation formelle des politiques de sécurité.

1.3 Les modèles de sécurité, petit historique

1.3.1 1971 – La matrice de droits d'accès de Lampson

Papiers du Pentagone

Les « papiers du Pentagone » est une expression désignant un document classé secret-défense (47 volumes, 7 000 pages) du département de la défense américaine, contenant des informations confidentielles à propos de l'implication politique et militaire des États-Unis dans la guerre du Viêt Nam de 1945 à 1971.

C'est dans l'objectif de spécifier de façon formelle les politiques de sécurité qu'ont été proposés les premiers modèles de sécurité. C'est en 1971, alors que le New York Times publiait des extraits de documents confidentiels des « papiers du Pentagone » sur la guerre du Viêt Nam (photocopiés et transmis par des fonctionnaires ayant accès au dossier), que Lampson, dans son papier qui fait désormais référence, « Protection » [Lampson, 1971], présentait une première formalisation de la notion de politiques de sécurité en proposant une catégorisation des différentes entités abstraites impliquées dans le processus de sécurité. Il disait en substance la chose suivante [2]. Tout système informatique peut être décrit au moyen d'un système abstrait appelé système d'objets composé des trois éléments fondamentaux suivants : un ensemble d'objets, un ensemble de sujets [3] et une matrice (ou fonction) d'accès. Les objets représentent les éléments du système informatique qui doivent être protégés. Des objets typiques sont les processus, les fichiers, les segments et les terminaux. Comme tout ce qui doit être spécifié, le choix des objets est une question de convention et dépend des besoins de protection de chaque système. Les sujets sont les entités qui doivent accéder aux objets. Le principe fondamental relatif à la notion de sujets est la possibilité d'envisager que les sujets puissent avoir des droits d'accès différents les uns des autres

2. Traduction libre et vocabulaire « actualisé ».

3. Dans son article, Lampson parle de « domaines » mais dans un souci d'uniformisation de notre présentation et pour être conforme à la littérature, nous nous permettons de renommer cette notion en « sujets ».

sur un ensemble d'objets partagés. Bien que naturelle pour le lecteur contemporain lisant ces lignes, cette idée était pourtant à l'époque relativement novatrice en ce sens où certains systèmes étaient composés de processus qui ne partageaient rien et ne communiquaient entre eux qu'au moyen de messages. La notion de partage était alors binaire, un objet étant soit exclusif, soit entièrement partagé. Lampson proposait donc une notion de partage plus fine. L'accès des sujets aux objets est déterminé par la matrice d'accès. Les lignes de cette matrice sont étiquetées par les noms de sujets et ses colonnes sont étiquetées par les noms des objets. L'élément de la matrice situé à l'intersection de la ligne correspondant au sujet s et la colonne étiquetée par l'objet o spécifie l'ensemble des modes dans lesquels le sujet s peut accéder à l'objet o. Les modes d'accès évoqués dans le papier original sont « read », « write », « call », « owner », « wakeup », ... Le premier modèle de sécurité était né. Le modèle de Lampson a été affiné l'année suivante par Graham et Denning [Graham and Denning, 1972].

1.3.2 1973 – Le modèle de Bell et LaPadula

Deux années plus tard et deux mois avant que n'éclate le scandale du Watergate, Bell et LaPadula proposaient une formalisation de la politique de sécurité multiniveau du Département de la Défense des États-Unis. Ils proposèrent dans [Bell and LaPadula, 1973] un modèle, aujourd'hui appelé BLP pour Bell et LaPadula, destiné à assurer la propriété de confidentialité. Comme dans le modèle proposé par Lampson, le modèle proposé par Bell et LaPadula est basé sur les notions de sujets, d'objets et de modes d'accès mais l'objectif de sécurité s'exprime en terme de flux d'information et non plus en terme d'accès. Le système est spécifié au moyen d'un système de transitions dont chaque état représente les accès courants. Formellement, si S représente l'ensemble des sujets, O celui des objets et A celui des modes d'accès ($A = \{read, write\}$), un état du système est un élément M de $\wp(S \times O \times A)$. Dans ce modèle est également introduite la notion de niveaux de sécurité formant un ensemble partiellement ordonné $(L, \leq)$. Tous les sujets recoivent un niveau de sécurité, appelé niveau d'habilitation, pour pouvoir accéder au système d'information. Tous les objets gérés par le système reçoivent également un niveau de sécurité, appelé niveau de classification. La fonction d'habilitation est notée $f_S : S \to L$ et celle de classification $f_O : O \to L$. Il est admis dans le modèle que l'attribution des niveaux de sécurité ne peut être modifié une fois établie (propriété de tranquilité forte). Le calcul des autorisations repose sur la vérification des deux propriétés suivantes.

- La propriété de sécurité simple : un sujet s ne peut lire un objet o seulement si son niveau d'habilitation est supérieur ou égal au niveau de classification de l'objet. Formellement : $\forall s \in S, \forall o \in O, (s, o, read) \in M \Rightarrow f_S(s) \geq f_O(o)$. Cette propriété est aussi connue sous le nom du principe de « no read up ».
- La $\star$-propriété : un sujet s ne peut écrire dans un objet o seulement si son niveau d'habilitation est inférieur ou égal au niveau de classification de o. Formellement : $\forall s \in S, \forall o \in O, (s, o, write) \in M \Rightarrow f_S(s) \leq f_O(o)$.

Si la première propriété formalise assez naturellement la notion de confidentialité de l'information, la seconde est motivée par la volonté de se prémunir des attaques par chevaux de Troie. Rappelons qu'un cheval de Troie est une application conçue pour

exécuter des actions à l'insu de l'utilisateur ayant exécuté l'application. L'objectif d'un cheval de Troie est généralement d'utiliser les droits de l'utilisateur pour détourner, diffuser ou détruire des informations. La seconde propriété est donc basée sur la remarque suivante : si on estime qu'une personne habilitée Top-secret ne va pas d'elle-même diffuser des informations classées Top-secret dans un document public, un programme qu'elle aurait exécuté pourrait à son insu effectuer ce type d'action si les droits de l'utilisateur le lui permettaient. C'est la raison pour laquelle Bell et LaPadula ont proposé de ne pas accorder le droit d'écriture à un sujet sur un objet qui lui est inférieurement classé. Ils se sont ensuite attachés à démontrer que les flots d'informations engendrés par les actions de lecture et d'écriture préservaient la confidentialité de l'information quelle que soit la séquence d'action exécutée.

1.3.3 1976 – Le modèle de Harrison, Ruzzo et Ullman

Cinq années après la publication du modèle proposé par Lampson, Harrison, Ruzzo et Ullman se sont intéressés aux problèmes relatifs à l'administration des politiques basées sur les matrices de droits d'accès. Ils ont développé un modèle, désormais appelé modèle HRU, basé sur une description du système par une matrice de droits d'accès et d'un ensemble fini de règles d'administration. Ces règles décrivent les conditions dans lesquelles il est permis de modifier le contenu de la matrice d'accès et sont décrites au moyen de commandes dont la syntaxe est la suivante : $if\ C\ then\ OP$ où C est une conjonction de conditions d'appartenance de droits dans l'état courant de la matrice d'accès et OP est une séquence d'opérations (donner/retirer un droit r à un sujet s sur un objet o et créer/détruire un sujet/objet). La question qu'ils ont investiguée est la suivante : étant donnée une matrice de droits d'accès initiale, est-il possible de déterminer si une séquence de commandes peut engendrer l'apparition d'un droit donné dans une cellule de la matrice dans laquelle il n'était pas présent initialement ? Ce problème, appelé problème de sûreté, consiste à déterminer si un sujet est capable, relativement à un ensemble de commandes donné, d'obtenir un droit sur un objet qu'il ne possédait pas sans que le propriétaire de l'objet le lui ait accordé. Ils ont établi des résultats de décidabilité sur la question en fonction de la forme des commandes. En particulier, ils ont établi que le problème de sûreté était décidable dans le cas d'un système dans lequel toutes les commandes sont caractérisées par l'application d'une seule opération. Ils ont également prouvé que le problème était indécidable en général.

1.3.4 1996 – Role Based Access Control

Une des limitations des modèles basés sur les matrices de droits d'accès est la nécessité d'exprimer de façon extensionnelle (*i.e.* énumérer) l'ensemble des autorisations pour chaque sujet, action et objet. La politique de sécurité devient donc rapidement complexe à exprimer et à administrer, en particulier lorsqu'un nouveau sujet ou objet est créé, ce qui nécessite de définir les nouveaux droits associés à ce sujet ou cet objet. Pour pallier ce problème, des modèles offrant la possibilité de structurer les sujets et objets autour de concepts plus abstraits et permettant d'établir un schéma de calcul des autorisations plus complexe que la simple vérification

d'un droit dans une matrice ont été explorés. C'est en particulier ce qu'a proposé Sandhu dans [Sandhu et al., 1996] lorsqu'il préconisait la structuration des sujets autour du concept de rôle. Le modèle proposé correspondant, appelé RBAC (Role Based Access Control), consiste à associer les autorisations aux rôles plutôt qu'aux sujets eux-mêmes. Dans RBAC, les rôles recoivent un ensemble de privilèges pour chaque objet et chaque sujet est affecté à un ensemble de rôles. Un autre concept introduit dans ce modèle est celui de session. Pour pouvoir réaliser une action sur un objet, un sujet doit préalablement initier une session et activer ensuite un des rôles qu'il possède lui permettant de réaliser cette action. Dans le cas général, un sujet peut activer tous les rôles qu'il souhaite au cours d'une session (sous réserve que les rôles en question lui aient été affectés). Le modèle RBAC introduit alors la notion de contrainte permettant de spécifier les conditions de compatibilité d'activation des rôles au sein d'une même session. Dans ce modèle, il est également possible de strucurer les rôles selon une hiérarchie, permettant de simplifier encore la gestion des autorisations. Le modèle RBAC est sans nul doute aujourd'hui un standard parmi tous les modèles de politiques de sécurité développés. Il a d'ailleurs servi de base à de nombreuses extensions.

1.4 Positionnement

Nous venons de présenter les premiers résultats émanant d'une volonté de formalisation et de factorisation des efforts à fournir en termes de conception de politiques de sécurité. Les modèles de sécurité fournissent des cadres plus ou moins génériques permettant de mettre en place une politique de sécurité au sein d'un système. Les modèles constituent en quelque sorte le « prêt-à-porter » du concepteur de politiques de sécurité. Pour autant, ils ne permettent pas nécessairement de raisonner de façon automatique sur les politiques qu'ils encadrent. Ceci est dû au fait que les modèles s'attachent surtout à définir les notions et les concepts de sécurité à partir desquels les politiques peuvent être définies. En revanche, les modèles ne sont pas nécessairement liés à un formalisme ni à un langage de spécification particulier. L'émergence des modèles de sécurité constitue une première étape vers une approche formelle de la sécurité mais il est nécessaire, afin de pouvoir automatiser le raisonnement sur les politiques décrites, de proposer aux concepteurs de politiques non pas des modèles mais des langages de spécification dont la sémantique s'exprime au moyen d'objets formels. Nos travaux s'inscrivent dans une démarche d'élaboration d'un cadre formel dont l'objectif est de spécifier des politiques de sécurité d'une part, et de permettre l'application de méthodes de vérification automatique sur ces spécifications d'autre part. Un cadre formel se bâtit à partir de deux principaux ingrédients : un langage de spécification et un système de vérification. Ces deux composantes sont très inégalement développées suivant les approches et les outils [Monin, 2000].

Pour distinguer les approches formelles dédiées aux politiques de sécurité proposées dans la littérature, nous établissons deux axes de comparaison : l'échelle de généricité et l'échelle d'automatisation. Sur le plan de la généricité, une partie de nos travaux s'inscrit dans une démarche générique, c-à-d non dédiée à l'étude d'une catégorie de politiques, et une autre partie de nos travaux se focalise sur un domaine spé-

cifique (les politiques de sécurité réseaux). Concernant les aspects « automatisation », nous nous distinguons des travaux présentant des cadres à vocation purement descriptive ou dans lesquels le raisonnement se fait manuellement. Dans le dernier cas, les cadres élaborés sont qualifiés de « cadres sémantiques » en ce sens où le raisonnement s'y effectue sur des structures sémantiques (automates, ensembles, relations, ...). Ce qui les distinguent notablement de notre approche est l'absence d'un langage de spécification et plus généralement l'absence de considérations syntaxiques. Parmi ces travaux, citons [Schneider, 2000, Morisset, 2007, Habib et al., 2009, Jaume, 2010].

1.5 Feuille de route et contributions

Chapitre 2 : Fondements techniques

Le chapitre 2 présente les notions nécessaires à la compréhension des différentes contributions présentées dans ce manuscrit. Deux mots traversent ce chapitre de bout en bout : syntaxe et sémantique. Dans un premier temps, les notions relatives à l'algèbre des termes sont présentées. Les propriétés liées aux relations binaires (terminaison, confluence, ...), les systèmes de réécriture puis les langages réguliers d'arbres (ou de termes) sont ensuite abordés. Les définitions essentielles de la logique des prédicats sont enfin présentées. Le chapitre se termine par une discussion sur les liens entretenus par les systèmes de réécritures et les automates d'arbres avec la logique du premier ordre.

Chapitre 3 : Un cadre formel pour la spécification, l'analyse et la transformation de politiques de sécurité

Le travail présenté dans le chapitre 3 s'inscrit dans une démarche de développement d'un cadre formel pour spécifier, analyser et maintenir les politiques de sécurité. Nous y montrons en quoi les spécifications des politiques de sécurité en deux parties sont particulièrement attractives, notamment en terme de réutilisabilité des spécifications. Dans ce cadre, nous définissons les politiques de sécurité comme des combinaisons de modèles de sécurité et de configurations. Les modèles, comme nous l'avons évoqué, ont pour objectif de formaliser les éléments de sécurité en fonction desquels les autorisations vont être déterminées ainsi que le processus de calcul desdites autorisations. La configuration consiste, quant à elle, à donner un sens aux symboles définis par le modèle. Nous nous distinguons de la plupart des travaux existants par l'utilisation de deux langages différents pour exprimer les modèles de sécurité et les configurations. Les modèles de sécurité sont décrits au moyen de contraintes du premier ordre tandis que les configurations sont spécifiées au moyen de programmes logiques dits programmes $\mathcal{H}_1$. Ainsi, nous proposons une démarche de spécification hétérogène permettant d'être conforme aux véritables besoins de description de chaque composant tout en permettant la combinaison de ces spécifications à des fins de vérification. Nous montrons que la spécification séparée de ces composants permet d'aborder la problématique intéressante et néanmoins peu (voire pas) traitée dans la littérature de la conversion de politiques d'un modèle vers un autre. Ce problème consiste à déterminer, étant donnés un modèle et une configu-

ration d'origine ainsi qu'un modèle cible, une configuration qui, combinée au modèle cible, engendre exactement les mêmes autorisations que le modèle et la configuration d'origine.

Chapitre 4 : Un cadre formel pour la spécification et la vérification de politiques de sécurité dynamiques

Le cadre présenté dans le chapitre 3 est basé sur une vision statique des politiques de sécurité. Nous nous proposons dans ce chapitre de pallier cette limitation en intégrant les effets de bords induits par l'exécution des actions. Plus précisément, on envisage dans cette partie de nos travaux les politiques de sécurité comme un moyen de restreindre l'évolution d'un programme. Un programme est alors constitué d'un certain nombre d'actions dont l'application modifie l'état courant du système qu'il décrit. Les états du systèmes sont vus comme des interprétations logiques finies (ensembles finis de faits) et les transitions sont décrites au moyen de règles de mises à jour des interprétations. Les politiques sont exprimées au moyen de règles de réécriture contraintes par des formules du premier ordre. Nous définissons ensuite la notion de système sécurisé, matérialisant le résultat de l'application d'une politique de sécurité sur un système. Nous proposons dans un second temps de décrire la sémantique opérationnelle des spécifications décrites dans notre cadre au moyen de systèmes de réécriture possédant des propriétés particulières. Notre approche se distingue de la plupart des approches formelles proposées pour la spécification des politiques dans la littérature dans le sens où le formalisme de spécification est différent du formalisme de raisonnement. Ce choix nous permet notamment d'imposer des restrictions techniques de façon totalement transparente pour l'utilisateur. Nous montrons ensuite en quoi la forme particulière des systèmes de réécriture manipulés permet d'utiliser les outils et techniques existants pour analyser les politiques de sécurité : étude des propriétés (cohérence, terminaison, complétude), du problème de l'administration, comparaison des politiques et analyse par requêtage.

Chapitre 5 : Les politiques de sécurité dans les réseaux informatiques

Nous proposons dans ce chapitre de modéliser les politiques de sécurité réseaux sous la forme de systèmes de réécriture et d'automates d'arbres. Nous montrons que cette approche permet d'effectuer plusieurs types d'analyse sur les firewalls isolés ainsi que sur leurs combinaisons. En particulier, nous montrons que les spécifications proposées sont adaptées à la vérification des propriétés sémantiques des firewalls ainsi qu'à leur comparaison. Nous montrons ensuite que la modélisation permet de détecter les anomalies de configuration et d'interroger les firewalls. Nous établissons une théorie décidable du premier ordre dans laquelle les anomalies et les requêtes peuvent s'exprimer, fournissant ainsi une méthode générant de façon automatique une procédure de détection des anomalies et de résolution de requêtes. Dans une seconde partie, nous nous focalisons sur la combinaison de firewalls, vus comme des politiques locales, introduisant les notions de topologie et de stratégie de routage. Nous montrons qu'il est possible d'obtenir automatiquement un système de réécriture simulant le traffic d'un réseau soumis à une politique de sécurité. La forme particulière

des règles obtenues est alors étudiée pour effectuer des vérifications de complétude, d'accessibilité, de terminaison et de cohérence.

Chapitre 6 : Synthèse

Le dernier chapitre récapitule les différentes contributions réalisées dans la présente thèse. Nous y discutons également des perspectives de travaux ensivageables.

2 Fondements techniques

Sommaire

2.1 Notations et définitions de base 22
2.1.1 Ensembles 22
2.1.2 Relations 22
2.1.3 Morphismes 24
2.1.4 Relations binaires 24
2.1.5 Induction 26
2.2 Algèbre des termes 27
2.2.1 Termes 27
2.2.2 Σ-algèbre 32
2.2.3 Spécification équationnelle 33
2.2.4 Problème du mot, raisonnement équationnel 35
2.3 Réécriture 36
2.3.1 Définitions 37
2.3.2 Preuves de propriétés 38
2.4 Automates finis d'arbres 39
2.4.1 Langages d'arbres 39
2.4.2 Langages de tuples 42
2.5 Logique du premier ordre 45
2.5.1 Syntaxe 45
2.5.2 Sémantique 47
2.5.3 Sémantiques syntaxiques et constructeurs 48

Ce chapitre présente les notions nécessaires à la bonne compréhension des travaux exposés dans cette thèse. Il est volontairement détaillé pour permettre aux lecteurs appartenant à des communautés diverses de pouvoir apprécier le contenu de ce manuscrit. L'ensemble des définitions et propriétés de ce chapitre faisant désormais partie du patrimoine scientifique commun, le lecteur y trouvera peu de références. Cependant, rendons à César ce qui lui appartient, un certain nombre d'ouvrages ont inspiré la présentation du présent chapitre, les principaux étant [Marchand, 2003, Kirchner and Kirchner, 2006, Comon and Jouannaud, 2003, Baader and Nipkow, 1998].

2.1 Notations et définitions de base

Cette section a pour objectif de fixer le vocabulaire utilisé par la suite. Nous y présentons les définitions relatives à la notion de relation ainsi que le principe d'induction. Le lecteur averti peut sans encombre se dispenser de sa lecture.

2.1.1 Ensembles

Si E est un ensemble, alors on notera :
- $card(E)$ le cardinal de E, c.-à-d. le nombre d'éléments de E ;
- $\wp(E)$ ou 2^E l'ensemble des parties de E, c.-à-d. $\{F \mid F \subseteq E\}$;
- $\overline{E}^F$ le complémentaire de E dans F. Lorsque F est clairement déterminé par le contexte, alors on notera simplement $\overline{E}$;

Définition 2.1. Soit E un ensemble non vide. On appelle **partition** de E tout sous-ensemble P de $\wp(E)$ tel que pour tout élément p de P, $p \neq \varnothing$, pour tous éléments distincts p et q de P, $p \cap q = \varnothing$ et $\bigcup_{p \in P} p = E$.

Définition 2.2. Soit E un ensemble fini. Un **E-ensemble** est un ensemble dont chaque élément est associé à un élément de E. Si F est un E-ensemble, alors la famille des ensembles d'éléments de F associés à un même élément de E forme une partition de F.

Si $E, E_1, \ldots, E_n$ sont des ensembles, alors on note :
- $E_1 \times \ldots \times E_n$ ou $\prod_{i=1}^{n} E_i$ le produit cartésien des E_i ;
- E^n le produit $\underbrace{E \times E \times \ldots \times E}_{n \text{ occurences de } E}$;

On appelle n-uplet ou tuple (si l'on ne souhaite pas préciser n) un élément d'un produit cartésien de n ensembles.

2.1.2 Relations

On rappelle qu'une **relation** d'un ensemble E dans un ensemble F est un sous-ensemble $\mathcal{R}$ de $E \times F$. E et F peuvent être des produits cartésiens d'ensembles. Si $E = E_1 \times \ldots \times E_n$ et $F = F_1 \times \ldots \times F_m$, alors $\mathcal{R}$ est dite $(m+n)$-aire. Par abus de

langage, on dira que $\mathcal{R}$ est une relation sur $E_1 \times \ldots \times E_n \times F_1 \times \ldots \times F_m$ (bien que ce soit ambigu).

Si $\mathcal{R}$ est une relation, on note indifféremment $x\mathcal{R}y$ ou $(x, y) \in \mathcal{R}$ pour indiquer que x et y sont en relation par $\mathcal{R}$. Dans ce cas, on dit que y est une **image** de x par $\mathcal{R}$ ou encore que x est un **antécédent** de y par $\mathcal{R}$. La relation réciproque est notée $\mathcal{R}^{-1}$ et se définit par l'ensemble $\{(y, x) \in F \times E \mid (x, y) \in \mathcal{R}\}$.

Définition 2.3. Soit $\mathcal{R}$ une relation de E dans F. On appelle **domaine** de $\mathcal{R}$ l'ensemble $\mathcal{D}om(\mathcal{R}) = \{x \in E \mid \exists y, (x, y) \in \mathcal{R}\}$ et on appelle **codomaine** ou **image** de $\mathcal{R}$ l'ensemble $\mathcal{I}m(\mathcal{R}) = \mathcal{C}odom(\mathcal{R}) = \{y \in F \mid \exists x, (x, y) \in \mathcal{R}\}$. Pour tout sous-ensemble A de E, on appelle image de A par $\mathcal{R}$ et l'on note $\mathcal{R}(A)$ l'ensemble $\{y \in F \mid \exists x \in A, (x, y) \in \mathcal{R}\}$. Pour tout élément a de E, on notera $\mathcal{R}(a)$ en lieu et place de $\mathcal{R}(\{a\})$.

Si $\mathcal{R}$ est une relation de E dans F et A est un sous-ensemble de E, alors la **restriction** de $\mathcal{R}$ à A, c.-à-d. $\mathcal{R} \cap (A \times F)$, est notée $\mathcal{R}_{|A}$. Si $\mathcal{R}'$ est une restriction de $\mathcal{R}$, alors $\mathcal{R}$ est un **prolongement** de $\mathcal{R}'$.

Si $\mathcal{R}_1 \subseteq E \times F$ et $\mathcal{R}_2 \subseteq F \times G$ sont deux relations respectivement de E dans F et de F dans G, alors on note $\mathcal{R}_2 \circ \mathcal{R}_1$ la **composition** de $\mathcal{R}_1$ et $\mathcal{R}_2$, c.-à-d. la relation de E dans G définie par $\{(x, z) \in E \times G \mid \exists y \in F,\ (x, y) \in \mathcal{R}_1 \text{ et } (y, z) \in \mathcal{R}_2\}$.

Définition 2.4. Soit $\mathcal{R}$ une relation de E dans F. On dit que :
- $\mathcal{R}$ est une **relation fonctionnelle** ou une **fonction** si tout élément de E a au plus une image par $\mathcal{R}$.
- $\mathcal{R}$ est **totale** si tout élément de E possède une image par $\mathcal{R}$. Elle est dite **partielle** dans le cas contraire.

Une fonction totale est appelée **application**.

Lorsqu'une relation $\mathcal{R}$ de E dans F est une fonction, alors :
- on utilise les symboles f, g, ... préférentiellement à $\mathcal{R}$;
- on note $f(a) = b$ en lieu et place de $f(a) = \{b\}$;
- on note $f : E \to F$ préférentiellement à $f \subseteq E \times F$;
- le symbole $\mapsto$ est utilisé pour matérialiser la correspondance réalisée par une fonction.

Pour tout ensemble E, on note id_E l'application identité, c.-à-d. $id_E : x \in E \mapsto x$. Lorsque E est clairement déterminé par le contexte, on notera simplement id.

Définition 2.5. Une application f de E dans F est dite :
- **injective** (ou est une **injection**) si tout élément de F admet au plus un antécédent par f ;
- **surjective** (ou est une **surjection**) si tout élément de F admet au moins un antécédent par f ;
- **bijective** (ou est une **bijection**) si tout élément de F admet au exactement un antécédent par f.

2.1.3 Morphismes

On rappelle qu'une **structure** est un ensemble muni d'un certain nombre d'objets mathématiques (relations, fonctions, applications, ...). Par exemple, on notera $(E, \mathcal{R})$ la structure composée de l'ensemble E muni de la relation $\mathcal{R}$ sur $E \times E$.

Définition 2.6. Un **morphisme** ou **homomorphisme** est une application entre deux ensembles munis d'un même type de structure qui respecte cette structure.

Par exemple, un morphisme de $(E, \oplus_E)$ dans $(F, \oplus_F)$, où $\oplus_E$ et $\oplus_F$ sont respectivement des applications de $E \times E$ dans E et de $F \times F$ dans F, est une application f de E dans F telle que $f(a \oplus_E b) = f(a) \oplus_F f(b)$.

Définition 2.7. Un morphisme d'une structure dans elle-même est appelé **endomorphisme**.

Définition 2.8. Un morphisme $f : E \to F$ est appelé **isomorphisme** si il existe un morphisme $g : F \to E$ tel que $f \circ g = id_F$ et $g \circ f = id_E$.

Un isomorphisme est donc une bijection pour laquelle les relations entre les éléments de l'ensemble d'arrivée sont les mêmes que celles entre leurs antécédents respectifs (la structure algébrique est préservée).

Définition 2.9. Deux ensembles E et F munis du même type de structure algébrique sont dits **isomorphes** s'il existe un isomorphisme de E dans F.

2.1.4 Relations binaires

Définition 2.10. Une relation binaire est une relation d'**équivalence** ssi elle est réflexive, symétrique et transitive.

Si E est un ensemble muni d'une relation d'équivalence $\mathcal{R}$, alors la **classe d'équivalence** d'un élément a de E est l'ensemble des images de a par $\mathcal{R}$. L'ensemble des classes d'équivalence de E suivant $\mathcal{R}$, noté $E/\mathcal{R}$ et appelé **ensemble quotient** de E par $\mathcal{R}$, forme alors une partition de E.

Définition 2.11. Une relation binaire sur $E \times E$ est :
- un **ordre** (sur E) ssi elle est réflexive, antisymétrique et transitive.
- un **ordre strict** (sur E) ssi elle est irréflexive et transitive.

Les ordres seront généralement notés par les symboles $\leq, \leq_E, \preceq, \mathit{inf}, \ldots$ Une relation d'ordre inf sur E est dite **totale**, si pour tous a et b de E, soit $(a, b) \in \mathit{inf}$, soit $(b, a) \in \mathit{inf}$. Dans le cas contraire, inf est un **ordre partiel**. Le cas échéant, on

dit que a et b sont **incomparables** (sous-entendu par *inf*) et l'on note $a \bowtie b$ si *inf* ne contient ni (a, b) ni (b, a).

Les relations binaires seront parfois dénotées par des flèches. Si $\to$ est une relation binaire, alors on note :

- $\leftarrow$ la relation inverse de $\to$ (c.-à-d. $\to^{-1}$) ;
- $\longleftrightarrow$ sa fermeture symétrique ;
- $\xrightarrow{+}$ sa fermeture transitive ;
- $\xrightarrow{*}$ sa fermeture réflexive transitive.

John Barkley Rosser

John Barkley Rosser (6 décembre 1907 – 5 septembre 1989) était un logicien américain, étudiant d'Alonzo Church. Il est connu pour sa participation au théorème de Church-Rosser dans le lambda calcul. Il a également développé ce qui est désormais appelé le crible de Rosser en théorie des nombres.

Définition 2.12. Une relation $\to$ sur un ensemble E est dite :

- **confluente** ssi $\forall x, x_1, x_2 \in E$:

$$x \xrightarrow{*} x_1 \quad \wedge \quad x \xrightarrow{*} x_2 \quad \Rightarrow \quad \exists y \in E,\ x_1 \xrightarrow{*} y \quad \wedge \quad x_2 \xrightarrow{*} y$$

- **localement confluente** ssi $\forall x, x_1, x_2 \in E$:

$$x \to x_1 \quad \wedge \quad x \to x_2 \quad \Rightarrow \quad \exists y \in E,\ x_1 \xrightarrow{*} y \quad \wedge \quad x_2 \xrightarrow{*} y$$

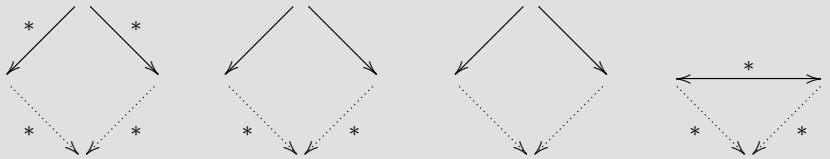
- **fortement confluente** ssi $\forall x, x_1, x_2 \in E$:

$$x \to x_1 \quad \wedge \quad x \to x_2 \quad \Rightarrow \quad \exists y \in E,\ x_1 \to y \quad \wedge \quad x_2 \to y$$

- possèdée la propriété de **Church-Rosser** ssi $\forall x_1, x_2 \in E$:

$$x_1 \xleftrightarrow{*} x_2 \quad \Rightarrow \quad \exists y \in E,\ x_1 \xrightarrow{*} y \quad \wedge \quad x_2 \xrightarrow{*} y$$

Ces différentes propriétés s'illustrent par les schémas suivants :



confluence *confluence locale* *confluence forte* *Church-Rosser*

Les flèches pleines représentant les hypothèses et les flèches en pointillés les conclusions.

Emmy Noether

Amalie Emmy Noether (23 mars 1882 – 14 avril 1935) était une mathématicienne allemande connue pour ses contributions révolutionnaires en algèbre abstraite et physique théorique. Décrite par Albert Einstein et d'autres comme la femme la plus importante de l'histoire des mathématiques, elle a révolutionné les théories des anneaux, des corps et des algèbres.

Remarque. Les différentes propriétés précédemment énoncées sont liées de la façon suivante :

$$\text{confluence forte} \ \Rightarrow \ \text{confluence} \ \Leftrightarrow \ \text{Church-Rosser} \ \Rightarrow \ \text{confluence locale}$$

Définition 2.13. On dit qu'une relation binaire $\to$ sur E est **noethérienne** ou **termine** ssi il n'existe pas de fonction $\varphi : \mathbb{N} \to E$ telle que $\forall n \in \mathbb{N}, \varphi(n) \to \varphi(n+1)$.

Définition 2.14. Soit $\rightarrow$ une relation binaire sur un ensemble E. Pour tout élément x de E, on dit que :

- x est **réductible** ssi $\exists y \in E,\ x \rightarrow y$, **irréductible** sinon,
- y est une **forme normale** de x ssi $x \xrightarrow{*} y$ et y irréductible.

Si x possède une unique forme normale, alors on la note $x{\downarrow}$.

Définition 2.15. Soit $\rightarrow$ une relation sur E. Si $\rightarrow$ termine, alors on a équivalence entre la propriété de Church-Rosser, la confluence et la confluence locale. Une relation noethérienne et confluente est dite **convergente** et dans ce cas, pour tous termes t et t', si $t \xleftrightarrow{*} t'$, alors $t{\downarrow} = t'{\downarrow}$.

Exemple 2.16. Soit $E = \{a, b, c, d\}$ et $\rightarrow$ la relation binaire sur E représentée par le graphe suivante :

$$c \longleftarrow a \rightleftarrows b \longrightarrow d$$

Les éléments c et d sont irréductibles et sont tous les deux une forme normale de a et de b. La relation $\rightarrow$ est localement confluente, non noethérienne (non terminante) et non confluente.
La relation binaire sur $\mathbb{N}$ définie par $\{(n, n+1) \mid n \in \mathbb{N}\}$ est quant à elle confluente mais non noethérienne tandis que la relation $\{(n, n+1) \mid \exists m \in \mathbb{N},\ n = 2 \times m\}$, qui associe à tout entier pair son successeur, est confluente et terminante.

2.1.5 Induction

Définir inductivement (ou par récurrence) un ensemble, c'est donner un schéma récursif de construction de ses éléments. Si E est défini inductivement, alors tous ses éléments peuvent s'obtenir à partir de règles de base en appliquant un nombre fini de fois des règles d'induction. Les définitions inductives sont souvent introduites de la façon suivante : « Soit E le plus petit ensemble vérifiant ... » ou plus simplement « Soit E l'ensemble défini inductivement par : ... » suivi de la donnée :

- d'un ou plusieurs cas de base ;
- d'une ou plusieurs règles d'induction.

On pourra également parler de plus petit ensemble fermé par un certain nombre d'opérations.

Par exemple, la fermeture réflexive transitive d'une relation $\mathcal{R}$ sur $E \times E$ peut être définie de la façon suivante : $\mathcal{R}^*$ est la plus petite relation telle que :

- id_E est inclus dans $\mathcal{R}^*$;
- pour tous couples d'éléments $(a, b) \in \mathcal{R}^*$ et $(b, c) \in \mathcal{R}$, le couple (a, c) appartient à $\mathcal{R}^*$.

$\mathcal{R}^*$ est donc la plus petite relation contenant id_E et $\mathcal{R}$ et fermée pour l'opération de composition.

Il peut être utile de décrire les définitions inductives au moyen de règles d'inférence. C'est notamment le cas lorsque les conditions invoquées dans le processus

d'induction se formalise sous la forme d'une conjonction de conditions d'appartenance.

Définition 2.17. Une **règle d'inférence** est la donnée d'un ensemble (éventuellement vide) d'hypothèses $H_1, \ldots, H_n$, d'une conclusion C et éventuellement d'une étiquette l. On note généralement les règles d'inférence sous la forme d'une fraction :

$$\frac{H_1 \quad \ldots \quad H_n}{C}\ (l)$$

Un système d'inférence est un ensemble de règles d'inférence.

Un système d'inférence est souvent représenté sous la forme d'un schéma de règles, c.-à-d. par des règles paramétrées représentant l'ensemble des règles obtenues en instanciant les paramètres.

Définition 2.18. Étant donnée un système d'inférence, on peut **déduire** ou **dériver** une conclusion C si et seulement si il existe une règle ayant C pour conclusion et dont toutes les hypothèses sont dérivables (ce qui est le cas en particulier si la règle ne possède pas d'hypothèse).

Les règles d'inférences peuvent être utilisées pour décrire un processus de déduction ou de construction d'un ensemble. En particulier, les ensembles définis inductivement peuvent être spécifiés au moyen de règles d'inférences. Par exemple, la définition de la fermeture transitive réflexive d'une relation $\mathcal{R}$ peut être spécifiée par le système d'inférence suivant :

$$\frac{a \in E}{(a,a) \in \mathcal{R}^*}\ \text{(réflexivité)} \qquad \frac{(a,b) \in \mathcal{R}^* \qquad (b,c) \in \mathcal{R}}{(a,c) \in \mathcal{R}^*}\ \text{(transitivité)}$$

2.2 Algèbre des termes

Pour raisonner, déduire ou calculer, les méthodes considérées dans ce manuscrit se fondent sur la description des objets de l'étude au moyen de termes algébriques. On parle alors de représentation algrébrique ou symbolique. Ces termes, parfois appelés arbres, constituent la brique de base syntaxique des méthodes dites algébriques et jouent un rôle fondamental dans la mécanisation du raisonnement.

2.2.1 Termes

Pour construire les termes, on se dote d'un vocabulaire contenant l'ensemble des symboles que l'on est autorisé à utiliser ainsi qu'un schéma de composition de ces symboles. Les symboles seront désignés sous le nom de symboles de fonction, ou symboles fonctionnels, et le schéma de composition de ces symboles est donné au moyen de sortes et de profils. La donnée de toutes ces informations est appelée signature.

Définition 2.19. On appelle **signature** tout couple $\Sigma = (\mathcal{S}, \mathcal{F})$ où :

- $\mathcal{S}$ est un ensemble non vide dont les éléments sont appelés **sortes**
- $\mathcal{F}$ est un ensemble non vide, disjoint de $\mathcal{S}$, dont les éléments sont appelés **symboles de fonction**. À tout élément $\mathbf{f} \in \mathcal{F}$ est associée une suite non vide $(s_1, \ldots, s_n, s)$ d'éléments de $\mathcal{S}$ appelée **profil**. On note alors :

$$\mathbf{f} \ : \ \mathsf{s_1} \times \ldots \times \mathsf{s_n} \to \mathsf{s}$$

L'entier n est appelé **arité** de $\mathbf{f}$ et est noté $ar(\mathbf{f})$. La sorte s est appelée **co-arité** de $\mathbf{f}$. Un symbole d'arité nulle est appelé **constante**.

Remarque. L'ensemble des symboles de fonction d'arité n est noté $\mathcal{F}_n$. On a naturellement $\mathcal{F} = \bigcup_{n \geq 0} \mathcal{F}_n$.

Exemple 2.20. Considérons un exemple dans lequel on souhaite représenter les notions usuelles d'entiers (naturels) et de pile d'entiers. Pour décrire un ensemble au moyen d'une signature, il est nécessaire de se donner une définition inductive de l'ensemble considéré. Ici, on cherche à construire deux ensembles. À chaque ensemble faisons correspondre une sorte. Choisissons Nat pour les entiers naturels et Stack pour les piles d'entiers. Une définition inductive des entiers est la suivante : zéro est un entier et le successeur de tout entier est également un entier. Chaque cas d'induction induit la définition d'un symbole fonctionnel ainsi que d'un schéma de composition. Choisissons **zero** pour représenter symboliquement le chiffre zéro. La caractérisation de l'assertion « zéro est un entier » au niveau de la signature se fait par la déclaration du symbole **zero** comme étant de profil (Nat). L'assertion « tout successeur d'un entier est un entier » va se traduire quant à elle par la définition d'un symbole **succ** associé au profil $(\mathsf{Nat}, \mathsf{Nat})$ ou $\mathsf{Nat} \to \mathsf{Nat}$ indiquant que le symbole fonctionnel **succ** s'applique à un entier et représente un autre entier qui est son successeur. Le raisonnement pour les listes d'entiers est similaire. Une pile d'entiers est soit une pile vide, que l'on peut noter $\mathbf{nil_{Nat}}$, soit un élément constitué d'un entier et d'une pile assemblés au moyen d'un symbole de notre choix, ici $\mathbf{cons_{Nat}}$. La description des entiers et des piles d'entiers s'effectue donc au moyen de la signature $\Sigma = (\mathcal{S}, \mathcal{F})$ suivante :

- $\mathcal{S} = \{\mathsf{Nat}, \mathsf{Stack}\}$
- $\mathcal{F} = \begin{cases} \mathbf{zero} : & & \to & \mathsf{Nat} \\ \mathbf{succ} : & \mathsf{Nat} & \to & \mathsf{Nat} \\ \mathbf{nil_{Nat}} : & & \to & \mathsf{Stack} \\ \mathbf{cons_{Nat}} : & \mathsf{Nat} \times \mathsf{Stack} & \to & \mathsf{Stack} \end{cases}$

Définition 2.21. Soit $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature. On appelle **ensemble de variables** et on note $\mathcal{X}$ tout $\mathcal{S}$-ensemble dénombrable, disjoint de $\mathcal{F}$. Pour toute variable x de $\mathcal{X}$ et toute sorte s de S, la notation $x : \mathsf{s}$ signifie que la variable x est associée à la sorte s. Dans ce cas, on dit que x est de sorte s.

Définition 2.22. Soient $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature et $\mathcal{X}$ un $\mathcal{S}$-ensemble de variables. On appelle **ensemble des termes sur Σ et $\mathcal{X}$ sorté par $\mathsf{s} \in \mathcal{S}$** et on note $\mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s}}$ le plus petit ensemble défini inductivement de la façon suivante :

- tout élément $x \in \mathcal{X}$ tel que $x : \mathsf{s}$ appartient à $\mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s}}$
- pour tout symbole $\mathbf{f} : \mathsf{s_1} \times \ldots \times \mathsf{s_n} \to \mathsf{s} \in \mathcal{F}$ et pour tous termes $t_1 \in \mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s_1}}, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s_n}}$, $f(t_1, \ldots, t_n)$ appartient à $\mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s}}$.

Lorsqu'un terme t appartient à $\mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s}}$, on dira que t est sorté par s. Enfin, on appelle **ensemble des termes sur Σ et $\mathcal{X}$** et l'on note $\mathcal{T}(\Sigma, \mathcal{X})$ le $\mathcal{S}$-ensemble $\bigcup_{\mathsf{s} \in \mathcal{S}} \mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s}}$.

On introduit la possibilité de définir une relation d'ordre sur les sortes d'une signature. Dans ce cas, $\mathsf{s} \leq \mathsf{s}'$ si et seulement si $\mathcal{T}(\Sigma)_{\mathsf{s}} \subseteq \mathcal{T}(\Sigma)_{\mathsf{s}'}$. On parle alors de signature ordo-sortée. L'intérêt est de pouvoir définir des sortes plus « génériques » ou *a contrario* plus « précises » que d'autres (dans le même esprit que la notion d'héritage en programmation orientée objet).

Définition 2.23. Soient $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature et $\mathcal{X}$ un $\mathcal{S}$-ensemble de variables. On définit l'application $\mathcal{V}ar : \mathcal{T}(\Sigma, \mathcal{X}) \to \wp(\mathcal{X})$ qui à tout terme t associe l'**ensemble des variables** qui ont au moins une occurence dans t :

- $\forall x \in \mathcal{X},\ \mathcal{V}ar(x) = \{x\}$
- $\forall \mathbf{f}(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{X}) = \bigcup_{i=1}^{n} \mathcal{V}ar(t_i)$

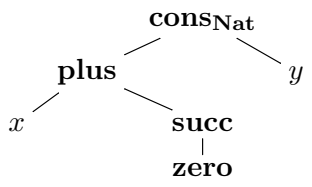
Un terme t est dit **clos** si $\mathcal{V}ar(t) = \varnothing$ et **linéaire** si les éléments de $\mathcal{V}ar(t)$ ont exactement une occurence dans t.

Remarque. On prolonge $\mathcal{V}ar$ à $\wp\big(\mathcal{T}(\Sigma, \mathcal{X})\big)$ de la façon suivante :

$$\forall \mathcal{E} \subseteq \mathcal{T}(\Sigma, \mathcal{X}),\ \mathcal{V}ar(\mathcal{E}) = \bigcup_{t \in \mathcal{E}} \mathcal{V}ar(t)$$

Remarque. L'ensemble des termes clos sur une signature $\Sigma = (\mathcal{F}, \mathcal{S})$ est noté $\mathcal{T}(\Sigma)$ au lieu de $\mathcal{T}(\Sigma, \varnothing)$. Sa définition peut être obtenue en modifiant la base de la définition inductive en remplaçant les variables par les constantes.
Remarque. Comme pour tout ensemble défini inductivement, on associe à chaque élément t de $\mathcal{T}(\Sigma, \mathcal{X})$ un arbre représentant les étapes de construction de t où chaque feuille est un élément de la base et chaque noeud représente une étape inductive.

Exemple 2.24. Soient la signature définie dans l'exemple 2.20 augmentée du symbole $\mathbf{plus} : \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat}$ et $\mathcal{X} = \{x : \mathsf{Nat},\ y : \mathsf{Stack}\}$ un $\mathcal{S}$-ensemble de variables. $\mathbf{cons_{Nat}}(\mathbf{plus}(x, \mathbf{succ}(\mathbf{zero})), y) \in \mathcal{T}(\Sigma, \mathcal{X})$ est représenté par l'arbre suivant :



Certaines sortes et certains symboles apparaissent de façon récurrence dans les spécifications algébriques. C'est la cas notamment des « constructeurs de listes ». Considérons une signature comportant une certaine sorte s. On supposera parfois implicitement l'existence de la sorte sList ainsi que des symboles suivants :

$$\begin{cases} \mathbf{nil_s} : & & \rightarrow & \mathsf{sList} \\ \mathbf{cons_s} : & \mathsf{s} \times \mathsf{sList} & \rightarrow & \mathsf{sList} \end{cases}$$

Pour faciliter l'écriture des termes de sorte sList, le symbole $\mathbf{nil_s}$ sera parfois remplacé par le symbole $[\,]$ et $\mathbf{cons_s}$ par ::.

Définition 2.25. Soit $\mathbb{N}_+$ l'ensemble des entiers naturels non nuls. Le monoïde $(\mathbb{N}_+^*,\ .\ )$ est le plus petit ensemble contenant les éléments de $\mathbb{N}_+ \cup \{\varepsilon\}$ clos pour l'opération (appelée concaténation) $.$ d'élément neutre ε.

Définition 2.26. Pour tout terme $t \in \mathcal{T}(\Sigma, \mathcal{X})$, on appelle **ensemble des positions** de t et on note $\mathcal{P}os(t)$ le plus petit sous-ensemble de $\mathbb{N}_+^*$ tel que :
- $\varepsilon \in \mathcal{P}os(t)$,
- si $t = \mathbf{f}(t_1, \ldots, t_n)$, alors pour tout i de $[1, n]$ et tout ω de $\mathcal{P}os(t_i)$, $i.\omega \in \mathcal{P}os(t)$.

Définition 2.27. On définit sur $\mathbb{N}_+^*$ la relation d'ordre partielle $\leq_{pref}$ par :

$$\forall \omega, \omega' \in \mathbb{N}_+^*,\ \omega \leq_{pref} \omega' \quad \Leftrightarrow \quad \exists \omega'' \in \mathbb{N}_+^*,\ \omega.\omega'' = \omega'$$

On dit alors que ω **est préfixe** de ω'.

Remarque. On dit que :
- ω est **préfixe strict** de ω' et on note $\omega <_{pref} \omega'$ ssi $\omega \leq_{pref} \omega'$ et $\omega \neq \omega'$
- ω et ω' sont **incomparables** et on note $\omega \bowtie \omega'$ ssi $\omega \not\leq_{pref} \omega'$ et $\omega' \not\leq_{pref} \omega$

Définition 2.28. Tout terme $t \in \mathcal{T}(\Sigma, \mathcal{X})$ peut être vu comme une application de $\mathcal{P}os(t)$ dans $\mathcal{F} \cup \mathcal{X}$ définie de la façon suivante :
- s'il existe une variable $x \in \mathcal{X}$ telle que $t = x$, alors $t(\varepsilon) = x$;
- s'il existe un symbole $\mathbf{f} \in \mathcal{F}$ et n termes $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})$ tels que $t = \mathbf{f}(t_1, \ldots, t_n)$, alors $\begin{cases} t(\varepsilon) = \mathbf{f} \\ \forall i \in [1, n],\ t(i.\omega) = t_i(\omega) \end{cases}$

On dit que $t(\varepsilon)$ est la racine ou le symbole de tête de t.

Remarque. Dans la littérature, il est parfois énoncé que la définition inductive de l'ensemble des termes construits sur une signature $\Sigma = (\mathcal{S}, \mathcal{F})$ est équivalente à celle de l'ensemble des fonctions de $\mathbb{N}_+^*$ dans $\mathcal{F} \cup \mathcal{X}$ respectant les profils de fonction (*i.e* les fonctions t telles que si $t(\omega) = \mathbf{f} : \mathbf{s_1} \times \ldots \times \mathbf{s_n} \rightarrow \mathbf{s}$ alors la co-arité du symbole $t(\omega.i)$ est $\mathbf{s_i}$). La seule différence entre ces deux définitions est que la première n'induit par construction que la définition de termes finis (c'est-à-dire ayant un nombre

fini de positions) tandis que la seconde permet la définition de termes infinis. Nous avons précédemment établi inductivement l'ensemble des termes construits sur une signature donnée et considérons donc dans la suite que tout terme est fini.

Définition 2.29. Soient $t \in \mathcal{T}(\Sigma, \mathcal{X})$ un terme et $\omega \in \mathcal{P}os(t)$ une position. On appelle **sous-terme de t à la position** ω et on note $t_{|\omega}$le terme tel que :

- si $\omega = \varepsilon$, alors $t_{|\omega} = t$;
- si $t = \mathbf{f}(t_1, \ldots, t_n)$, pour un certain $\mathbf{f} \in \mathcal{F}$ et certains termes $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})$, et $\omega = i.\omega'$ pour un certain $(i, \omega') \in \mathbb{N}_+ \times \mathbb{N}_+^*$, alors $t_{|\omega} = (t_i)_{|\omega'}$.

On dit qu'un terme t' est un **sous-terme** de t et on note $t' \trianglelefteq_{sub} t$ si il existe $\omega \in \mathcal{P}os(t),\ t' = t_{|\omega}$. On parle de sous-terme **strict** lorsque $\omega \neq \varepsilon$.

Définition 2.30. Soient $t, s \in \mathcal{T}(\Sigma, \mathcal{X})$ et $\omega \in \mathcal{P}os(t)$. On définit le terme noté $t\,[s]_\omega$ de la façon suivante :

- si $\omega = \varepsilon$, alors $t\,[s]_\omega = s$;
- si $t = \mathbf{f}(t_1, \ldots, t_n)$, pour un certain $\mathbf{f} \in \mathcal{F}$ et certains termes $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})$, et $\omega = i.\omega'$ pour un certain $(i, \omega') \in \mathbb{N}_+ \times \mathbb{N}_+^*$, alors $t\,[s]_\omega = \mathbf{f}(t_1, \ldots, t_i\,[s]_{\omega'}, \ldots, t_n)$.

On dit alors que **le sous-terme $t_{|\omega}$ a été remplacé par** s.

Nous tenons à introduire une notion qui n'apparaît pas de façon systématique dans les diverses présentations de l'algèbre des termes puisque la plupart du temps, les notions de remplacement et de position permettent d'y faire implicitement référence. Pourtant, nous le verrons un peu plus tard, cette notion est essentielle : il s'agit de la notion de contexte.

Définition 2.31. Un **contexte** est une application de $\mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s}}$ (pour une certaine sorte s) dans $\mathcal{T}(\Sigma, \mathcal{X})$ telle qu'il existe un terme $u \in \mathcal{T}(\Sigma)$ et une position $\omega \in \mathcal{P}os(u)$ tels que pour tout $t \in \mathcal{T}(\Sigma, \mathcal{X}), C(t) = u\,[t]_\omega$. Un tel contexte sera représenté par l'unique terme $\tilde{C}$ de $\mathcal{T}(\Sigma \cup \{\Box\})$ tel que pour toute position $\omega' <_{pref} \omega$ et $\omega' \bowtie \omega$, $\tilde{C}(\omega') = u(\omega)$ et $\tilde{C}(\omega) = \Box$. On confondra $\tilde{C}$ et C et l'on notera $C[t]$ à la place de $C(t)$. On notera $\mathcal{C}(\Sigma)$ l'ensemble des contextes sur Σ.

Remarque. L'ensemble des contextes sur Σ est isomorphe à l'ensemble des termes contenant exactement une variable. C'est pourquoi il nous arrivera parfois de « manipuler » de tels termes comme des contextes.

> *Exemple 2.32.* Soit Σ la signature définie dans l'exemple 2.24. Le terme $C = \mathbf{plus}(\Box, \mathbf{succ}(\mathbf{zero}))$ est un contexte sur Σ. Si t est le terme **zero**, alors $C[t]$ représente le terme $\mathbf{plus}(\mathbf{zero}, \mathbf{succ}(\mathbf{zero}))$. Le terme $\mathbf{plus}(x, \mathbf{succ}(\mathbf{zero}))$ sera parfois confondu avec C.

Définition 2.33. Soient $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature et $\mathcal{X}$ un $\mathcal{S}$-ensemble de variables. On appelle **substitution** (sous-entendu de $\mathcal{X}$ sur Σ) toute application

σ de $\mathcal{X}$ dans $\mathcal{T}(\Sigma, \mathcal{X})$. On appelle **domaine** de σ l'ensemble $\mathcal{D}om(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$ et **codomaine** de σ l'ensemble $\mathcal{I}m(\sigma) = \{\sigma(x) \mid x \in \mathcal{D}om(\sigma)\}$. Si σ est une bijection de $\mathcal{X}$ dans $\mathcal{X}$, alors σ est appelé renommage.

Remarque. On notera généralement $\{x \mapsto \sigma(x)\}_{x \in \mathcal{D}om(\sigma)}$ (c'est-à-dire extensionnellement) pour désigner la substitution σ.
Remarque. On dit qu'une substitution σ est **close** si $\mathcal{V}ar(\mathcal{I}m(\sigma)) = \varnothing$.

Définition 2.34. Soit $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})$ une substitution. On prolonge σ en un endomorphisme σ^* de $\mathcal{T}(\Sigma, \mathcal{X})$ de la façon suivante :

- pour tout $x \in \mathcal{X}$, $\sigma^*(x) = \sigma(x)$;
- pour tout terme $t = \mathbf{f}(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{X})$, $\sigma^*\big(t\big) = \mathbf{f}\big(\sigma^*(t_1), \ldots, \sigma^*(t_n)\big)$.

On confondra σ avec σ^*.

> *Exemple 2.35.* Soient Σ la signature définie dans l'exemple 2.24, t le terme $\mathbf{plus}(x, \mathbf{succ}(y))$ et σ la substitution $\{x \mapsto \mathbf{zero}\ ;\ y \mapsto \mathbf{succ}(z)\}$. Le terme $t' = \sigma(t)$ est donné par : $\mathbf{plus}(\mathbf{zero}, \mathbf{succ}(\mathbf{succ}(z)))$. On utilisera parfois une notation similaire à celle des contextes pour indiquer les variables contenues dans un terme. Par exemple, $t[x, y]$ nous informe que t a pour variables $\{x, y\}$ et le terme $t[t_1, t_2]$ représente le terme $\sigma(t)$ pour $\sigma = \{x \mapsto t_1\ ;\ y \mapsto t_2\}$. Dans notre exemple, $t' = t[\mathbf{zero}, \mathbf{succ}(z)]$.

Définition 2.36. On appelle **relation de subsomption** et l'on note $\preceq$ la relation telle que pour tous t et s de $\mathcal{T}(\Sigma, \mathcal{X})$, $t \preceq s$ ssi il existe une substitution $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})$ telle que $\sigma(t) = s$. On dit alors que t **subsume** s. Si ρ est un renommage des variables de t et de s telle que $\mathcal{V}ar(\rho(t)) \cap \mathcal{V}ar(\rho(s)) = \varnothing$ et que $\rho(t) \preceq \rho(s)$, alors on dit que t **filtre** s et l'on note $t \ll s$. Une substitution σ est dite **plus générale** que μ, ce que l'on note $\sigma \preceq \mu$ s'il existe une substitution σ' telle que $\mu = \sigma'_\circ\sigma$. On dit aussi que μ est une **instance** de σ.

Définition 2.37. Deux termes t et s sur une signature Σ sont dits **unifiables** s'il existe une substitution σ telle que $\sigma(t) = \sigma(s)$. Dans ce cas, σ est un **unificateur** de t et s. Si t et s sont unifiables, on appelle **unificateur principal** de t et s et l'on note $mgu(t, s)$ l'unificateur, unique à un renommage près, dont tout autre unificateur σ' de t et s est une instance.

2.2.2 Σ-algèbre

Jusqu'ici, les notions rencontrées étaient essentiellement syntaxiques. Nous avons défini un vocabulaire permettant de décrire des objets à partir de symboles fonctionnels. Ces symboles n'ont pas encore de « sens » à proprement parler. Pour pouvoir leur en attribuer un et énoncer des propriétés sur les objets ainsi décrits, il est nécessaire d'associer au vocabulaire de description une sémantique. Pour ce faire, l'on associe à chaque symbole fonctionnel un schéma de calcul appelé interprétation.

Définition 2.38. Soit $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature, on appelle Σ-algèbre (ou interprétation de Σ) $\mathcal{A}$ la donnée de :

- pour tout $\mathsf{s} \in \mathcal{S}$, un ensemble non vide $|\mathcal{A}|_{\mathsf{s}}$ appelé **domaine d'interprétation** des termes sortés par s,
- pour tout $\mathbf{f} : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \to \mathsf{s} \in \mathcal{F}$, une application

$$\mathbf{f}_{\mathcal{A}} : |\mathcal{A}|_{\mathsf{s}_1} \times \ldots \times |\mathcal{A}|_{\mathsf{s}_n} \to |\mathcal{A}|_{\mathsf{s}}$$

  appelée **interprétation de f**.

On note $\mathcal{A}lg(\Sigma)$ la classe des Σ-algèbres.

> *Exemple 2.39.* Soit $\Sigma = (\mathcal{S}, \mathcal{F})$ la signature comprenant la sorte Nat et les trois symboles fonctionnels **zero** $:\to \mathsf{Nat}$, **succ** : $\mathsf{Nat} \to \mathsf{Nat}$ et **plus** : $\mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat}$. La donnée de $|\mathcal{A}|_{\mathsf{Nat}} = \mathbb{N}$, $\mathbf{zero}_{\mathcal{A}} = 0$, $\mathbf{succ}_{\mathcal{A}} : x \mapsto x + 1$ et $\mathbf{plus}_{\mathcal{A}} : (x, y) \mapsto x + y$ forme une Σ-algèbre.

Remarque. $\mathcal{T}(\Sigma)$ est une Σ-algèbre appelée **algèbre syntaxique close** et définie de la façon suivante :

- pour tout $\mathsf{s} \in \mathcal{S}$, le domaine d'interprétation des termes sortés par s est l'ensemble des termes clos sortés par s : $\mathcal{T}(\mathcal{F})$,
- tout symbole de fonction $\mathbf{f} \in \mathcal{F}$ est trivialement interprété par lui-même : $\mathbf{f}_{\mathcal{T}(\mathcal{F})} = \mathbf{f}$.

Cette algèbre joue un rôle particulier puisqu'elle possède la propriété dite d'initialité : pour tout $\mathcal{A} \in \mathcal{A}lg(\Sigma)$, il existe un unique morphisme de $\mathcal{T}(\Sigma)$ dans $\mathcal{A}$.

Définition 2.40. Soient $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature et $\mathcal{A}$ une Σ-algèbre. On appelle **$\mathcal{A}$-valuation** toute application $\delta : \mathcal{X} \to \mathcal{A}$ telle que si $x : \mathsf{s}$, alors $\delta(x) \in |\mathcal{A}|_{\mathsf{s}}$.

Définition 2.41. Soient $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature, $\mathcal{A}$ une Σ-algèbre et δ une $\mathcal{A}$-valuation. On appelle **sémantique** dans $\mathcal{A}$ selon δ et on note $[\![.]\!]^{\delta}_{\mathcal{A}}$ le morphisme de $\mathcal{T}(\Sigma, \mathcal{X})$ vers $\mathcal{A}$ défini par :

- pour tout $x \in \mathcal{X}$, $[\![x]\!]^{\delta}_{\mathcal{A}} = \delta(x)$;
- pour tout terme $t = \mathbf{f}(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{X})$, $[\![t]\!]^{\delta}_{\mathcal{A}} = \mathbf{f}_{\mathcal{A}}\big([\![t_1]\!]^{\delta}_{\mathcal{A}}, \ldots, [\![t_n]\!]^{\delta}_{\mathcal{A}}\big)$.

2.2.3 Spécification équationnelle

On introduit dans cette section la notion de spécification équationnelle. Ce type de spécification consiste à établir un ensemble d'équations permettant de restreindre l'ensemble des interprétations « licites » des symboles fonctionnels. En définissant des propriétés d'égalité sur les termes de la signature, elles caractérisent une classe d'algèbres dans laquelle on se positionne pour raisonner et prouver des conjectures.

Définition 2.42. Soit $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature. On appelle **équation** ou **axiome** sur Σ (et sur un ensemble de variables $\mathcal{X}$) toute paire $(t, t') \in \mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s}} \times \mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s}}$, où $\mathsf{s} \in \mathcal{S}$, notée $t \cong t'$.

Définition 2.43. Soient $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature et $\mathcal{A}$ une Σ-algèbre. On dit que $\mathcal{A}$ est un **modèle** pour l'équation $t \cong t'$ ou encore que $t \cong t'$ est **valide** dans $\mathcal{A}$ et l'on note

$$\mathcal{A} \models t \cong t' \quad \text{ou} \quad t =_{\mathcal{A}} t'$$

si pour toute valuation δ, $[\![t]\!]^{\delta}_{\mathcal{A}} = [\![t']\!]^{\delta}_{\mathcal{A}}$.

Remarque. Soit E un ensemble d'équations. On dit que la Σ-algèbre $\mathcal{A}$ est un modèle de E et l'on note

$$\mathcal{A} \models E$$

si $\mathcal{A}$ est un modèle pour toutes les équations de E. Si $t \cong t'$ est une équation sur Σ, alors on notera également

$$E \models t \cong t'$$

si tout modèle de E est modèle de $t \cong t'$.

Remarque. Soient $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature et $\mathcal{A}$ une Σ-algèbre. On note $\mathcal{T}h(\mathcal{A})$ l'ensemble des équations valides dans $\mathcal{A}$ et $\mathcal{M}od(E)$ l'ensemble des Σ-algèbres modèles de E.

Définition 2.44. On appelle **spécification équationnelle** ou **spécification d'un type abstrait algébrique** tout couple (Σ, E) où Σ est une signature et E un ensemble d'équations sur Σ.

Remarque. Soient $\mathcal{A}$ une Σ-algèbre et $(\Sigma = (\mathcal{S}, \mathcal{F}), E)$ une spécification équationnelle. On dit que $\mathcal{A}$ est un modèle pour (Σ, E) si $\mathcal{A}$ est un modèle pour E.

Exemple 2.45. Soient $(\Sigma = (\mathcal{S}, \mathcal{F}), E)$ une spécification équationnelle et $\mathcal{X}$ un $\mathcal{S}$-ensemble de variables définis par :

- $\mathcal{S} = \{\mathsf{Bool}\}$
- $\mathcal{F} = \begin{cases} \mathbf{True} : & \to & \mathsf{Bool} \\ \mathbf{False} : & \to & \mathsf{Bool} \\ \neg : \mathsf{Bool} & \to & \mathsf{Bool} \\ \wedge : \mathsf{Bool} \times \mathsf{Bool} & \to & \mathsf{Bool} \\ \vee : \mathsf{Bool} \times \mathsf{Bool} & \to & \mathsf{Bool} \end{cases}$
- $\mathcal{X} = \{a : \mathsf{Bool}, b : \mathsf{Bool}\}$
- $E = \begin{cases} \neg\mathbf{True} & \cong & \mathbf{False} \\ \neg\mathbf{False} & \cong & \mathbf{True} \\ \mathbf{True} \wedge a & \cong & a \\ \mathbf{False} \wedge a & \cong & \mathbf{False} \\ \neg(a \wedge b) & \cong & \neg a \vee \neg b \\ \neg\neg a & \cong & a \end{cases}$

L'algèbre de Boole $\mathcal{B}$ définie par :

- $|\mathcal{B}|_{\mathsf{Bool}} = \{0, 1\}$
- $\mathbf{True}_{\mathcal{B}} = 1$
- $\mathbf{False}_{\mathcal{B}} = 0$
- $\wedge_{\mathcal{B}} = et : \{(0,0) \mapsto 0,\ (0,1) \mapsto 0,\ (1,0) \mapsto 0,\ (1,1) \mapsto 1\}$

George Boole

George Boole (2 novembre 1815 à Lincoln – 8 décembre 1864 à Ballintemple) est un logicien, mathématicien et philosophe britannique. Il est à l'origine de la logique fondée sur la structure algébrique qui porte son nom. Autodidacte, il publia ses premiers travaux d'algèbre tout en exerçant son métier d'instituteur et de directeur d'école dans la région de Lincoln. Il obtint la médaille royale de la Royal Society (académie des sciences britanniques) en 1844 puis une chaire de mathématiques au Queen's College de Cork en 1849.

- $\vee_{\mathcal{B}} = ou : \{(0,0) \mapsto 0,\ (0,1) \mapsto 1,\ (1,0) \mapsto 1,\ (1,1) \mapsto 1\}$
- $\neg_{\mathcal{B}} = non : \{0 \mapsto 1,\ 1 \mapsto 0\}$

est un modèle de (Σ, E).

2.2.4 Problème du mot, raisonnement équationnel

Définition 2.46. Soient $(\Sigma = (\mathcal{S}, \mathcal{F}), E)$ une spécification équationnelle, $t, t' \in \mathcal{T}(\Sigma, \mathcal{X})$, on appelle **problème du mot** associé à $t \cong t'$ la recherche de la validité de $t \cong t'$ dans tout modèle de E, autrement dit la démonstration de $E \models t \cong t'$.

Remarque. L'objectif du raisonnement équationnel est de ramener le problème du mot, qui est un problème sémantique, à un problème syntaxique.

Définition 2.47. Soit $(\Sigma = (\mathcal{S}, \mathcal{F}), E)$ une spécification équationnelle, on appelle **théorie équationnelle** et on note $=_E$ la plus petite congruence sur $\mathcal{T}(\Sigma, \mathcal{X})$ telle que :

$$\forall \sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X}),\ \forall (t \cong t') \in E,\ \sigma(t) =_E \sigma(t')$$

Autrement dit, par définition de la congruence sur une Σ-algèbre, $=_E$ est la fermeture symétrique réflexive transitive de la relation :

$$\{\big(C[\sigma(t)], C[\sigma(t')]\big) \mid \sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X}),\ (t \cong t') \in E,\ C \in \mathcal{C}(\Sigma)\}$$

Définition 2.48. Soit $(\Sigma = (\mathcal{S}, \mathcal{F}), E)$ une spécification équationnelle, on appelle **raisonnement équationnel** le raisonnement basé sur les règles d'inférences suivantes :

$$\frac{t \cong t' \in E}{t \cong t'}\text{(axiome)} \qquad \frac{}{t \cong t}\text{(réflexivité)}$$

$$\frac{t \cong t'}{t' \cong t}\text{(symétrie)} \qquad \frac{t \cong t' \quad t' \cong t''}{t \cong t''}\text{(transitivité)}$$

$$\frac{t \cong t'}{\sigma(t) \cong \sigma(t')} \forall \sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})\ \text{(substitution)} \qquad \frac{t \cong t'}{C[t] \cong C[t']} \forall C \in \mathcal{C}(\Sigma)\ \text{(contexte)}$$

On note $E \vdash t \cong t'$ si l'équation $t \cong t'$ peut être obtenue à partir de E en utilisant uniquement les règles du raisonnement équationnel.

Remarque. Les deux précédentes définitions sont équivalentes :

$$\forall t, t' \in \mathcal{T}(\Sigma, \mathcal{X}),\quad t =_E t' \quad ssi \quad E \vdash t \cong t'$$

Exemple 2.49. Soit (Σ, E) la spécification équationnelle décrite dans l'exemple 2.45. On a $\mathbf{False} \vee x =_E x$ comme le montre la dérivation suivante :

$$\cfrac{\cfrac{\cfrac{\neg\mathbf{True} \cong \mathbf{False} \in E}{\neg\mathbf{True} \cong \mathbf{False}}\ \text{(ax.)}}{\neg\mathbf{True} \vee x \cong \mathbf{False} \vee x}\ \text{(cont.)}}{\mathbf{False} \vee x \cong \neg\mathbf{True} \vee x}\ \text{(sym.)}$$

$$\dfrac{\dfrac{\dfrac{

d'un représentant unique par classe d'équivalence d'objets est à la base de bien des raisonnements destinés à prouver l'équivalence entre deux objets (minimalisation des automates, formes normales des fonctions booléennes, ...).

2.3.1 Définitions

On supposera dans la suite l'existence d'une signature $\Sigma = (\mathcal{S}, \mathcal{F})$ et d'un ensemble dénombrable de variables $\mathcal{X}$. La notion de terme fait donc référence aux éléments de $\mathcal{T}(\Sigma, \mathcal{X})$.

Définition 2.51. Une **règle de réécriture** (sous-entendu sur Σ et $\mathcal{X}$) est une paire de termes (l, r) de même sorte notée $l \to r$ telle que $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. l est le membre gauche de la règle et r le membre droit. Un **système de réécriture** est un ensemble de règles de réécriture.

Définition 2.52. Soit $\mathcal{R}$ un système de réécriture et t un terme. La **relation de réduction** engendrée par $\mathcal{R}$, notée $\to_{\mathcal{R}}$ est la plus petite relation contenant $\mathcal{R}$ stable par substitution et par contexte. Autrement dit, $t \to_{\mathcal{R}} t'$ ssi il existe :

- un contexte C ;
- une règle de réécriture $l \to r$ de $\mathcal{R}$;
- une substitution σ ;

tels que $t = C[\sigma(l)]$ et $t' = C[\sigma(r)]$. On dit alors que t est **réécrit** (ou se réécrit ou se réduit) en t' par $\mathcal{R}$ (en un pas). On note $=_{\mathcal{R}}$ la plus petite relation d'équivalence contenant $\to_{\mathcal{R}}$.

Remarque. En tant que relation binaire, les notations, définitions et propriétés de la section précédente s'appliquent à $\to_{\mathcal{R}}$. Par ailleurs, on confondra souvent un système de réécriture avec la relation de réduction qu'il engendre.

Par ailleurs, si p est une propriété sur les termes (être linéaire, être clos, ...), alors la règle de réécriture $l \to r$ vérifie :

- p à gauche si l vérifie p ;
- p à droite si r vérifie p ;
- p si l et r vérifient p.

Un système de réécriture vérifie une propriété p ssi toutes les règles qui le constituent la vérifie.

> *Exemple 2.53.* Soit Σ la signature définie dans l'exemple 2.39 augmentée du symbole $\mathbf{fibo} : \mathsf{Nat} \times \mathsf{Nat} \to \mathsf{Nat}$. Le système de réécriture $\mathcal{R}$ suivant :
>
> $$\begin{array}{rcl} \mathbf{zero} + x & \to & x \\ \mathbf{succ}(x) + y & \to & \mathbf{succ}(x + y) \\ \mathbf{fibo}(\mathbf{zero}) & \to & \mathbf{zero} \\ \mathbf{fibo}(\mathbf{succ}(\mathbf{zero})) & \to & \mathbf{succ}(\mathbf{zero}) \\ \mathbf{fibo}(\mathbf{succ}(\mathbf{succ}(x))) & \to & \mathbf{fibo}(x) + \mathbf{fibo}(\mathbf{succ}(x)) \end{array}$$
>
> est linéaire. Un exemple de réduction engendrée par $\mathcal{R}$ est donné par la sé-

quence suivante :

$$\begin{aligned}
\mathbf{fibo}(\underline{\mathbf{succ}(\mathbf{succ}(\mathbf{zero}))} + z) &\to_{\mathcal{R}} \mathbf{fibo}(\mathbf{succ}(\underline{\mathbf{succ}(\mathbf{zero}) + z})) \\
&\to_{\mathcal{R}} \mathbf{fibo}(\mathbf{succ}(\mathbf{succ}(\underline{\mathbf{zero} + z}))) \\
&\to_{\mathcal{R}} \underline{\mathbf{fibo}(\mathbf{succ}(\mathbf{succ}(z)))} \\
&\to_{\mathcal{R}} \mathbf{fibo}(z) + \mathbf{fibo}(\mathbf{succ}(z))
\end{aligned}$$

La partie soulignée de chaque terme (appelé radical ou redex) correspond au sous-terme filtrant la règle de réécriture appliquée.

2.3.2 Preuves de propriétés

L'utilisation des systèmes de réécriture a pour objectif, au delà de fournir un moyen opérationnel pour effectuer des preuves équationnelles, de permettre de prouver des propriétés sur les calculs qu'ils modélisent. Par exemple, lorsqu'un système de réécriture modélise le processus d'évaluation d'une fonction, les questions suivantes sont d'intêret :

- la fonction est-elle définie pour toutes les valeurs d'entrée possibles ?
- l'évaluation de la fonction fournit-elle pour toute entrée un résultat en temps fini ?
- les calculs sont-ils non ambiguë, autrement dit le système modélise-t-il bien une relation fonctionnelle ?

Par ailleurs, sous certaines conditions, un système de réécriture, vu comme la spécification inductive d'une relation, permet d'effectuer des preuves par induction.

Certaines propriétés d'un système de réécriture réfèrent aux propriétés correspondantes sur les relations de réduction qu'elles engendrent. En particulier, on dira qu'un système de réécriture est confluent, termine, ... ssi la relation de réduction associée est confluente, termine, ... D'autres propriétés ne s'expriment pas directement sur la relation de réduction engendrée. C'est le cas de la propriété de confluence sur les termes clos et de la complétude suffisante.

Définition 2.54. Soit $\mathcal{R}$ un système de réécriture sur une signature Σ. On dit que $\mathcal{R}$ est **confluent sur les termes clos** ssi la relation $\to_{\mathcal{R}} \cap\, \mathcal{T}(\Sigma) \times \mathcal{T}(\Sigma)$ est confluente.

Il s'agit en général de la propriété que l'on souhaite vérifier lorsque l'on s'intéresse à la non ambiguité des calculs modélisés par un système de réécriture. Un système peut être confluent sur les termes clos sans être confluent.

Exemple 2.55. Soit Σ une signature comportant les symboles $\mathbf{True} :\to \mathsf{Bool}$, $\mathbf{True} :\to \mathsf{Bool}$, $\mathbf{f} : \mathsf{Bool} \to \mathsf{Bool}$ et $\mathbf{g} : \mathsf{Bool} \to \mathsf{Bool}$. On définit le système de réécriture $\mathcal{R}$ suivant sur Σ :

$$\begin{aligned}
\mathbf{f}(\mathbf{True}) &\to \mathbf{True} \\
\mathbf{f}(\mathbf{False}) &\to \mathbf{True} \\
\mathbf{g}(x) &\to \mathbf{True} \\
\mathbf{g}(x) &\to \mathbf{f}(x)
\end{aligned}$$

> $\mathcal{R}$ est confluent sur les termes clos (tout terme clos se réduit en **True**) mais n'est pas confluent. En effet, le terme $\mathbf{g}(x)$ se réduit vers les deux termes irréductibles $\mathbf{f}(x)$ et **True**.

Deux autres propriétés essentielles sont appelées complétude suffisante et consistance. Ces propriétés sont relatives à un partitionnement de la signature en deux ensembles en fonction des rôles tenus par les symboles dans le processus de modélisation. Par exemple, si l'on considère la signature contenant les symboles **zero** $:\rightarrow \mathsf{Nat}$, **succ** : $\mathsf{Nat} \rightarrow \mathsf{Nat}$ et $+ : \mathsf{Nat} \rightarrow \mathsf{Nat}$, il semble évident que les symboles **zero** et **succ** ont pour objectif de décrire les éléments du discours (les entiers naturels) alors que le symbole + modélise une fonction des entiers vers les entiers. Dans ce cas, les symboles à vocation descriptive sont appelés symboles constructeurs tandis que les symboles modélisant des fonctions sont appelés symboles définis. Notons $\Sigma\mathscr{C}ons$ la partie de la signature contenant les symboles constructeurs et $\Sigma\mathscr{D}ef$ celle contenant les symboles définis. Nous reviendrons plus en détail sur ces notions en fin de chapitre. Les propriétés suivantes caractérisent le fait que les symboles constructeurs et les symboles définis tiennent bien les « rôles » de modélisation que nous venons d'évoquer.

Définition 2.56. Soit $\mathcal{R}$ un système de réécriture sur une signature Σ. On dit que $\mathcal{R}$ est **consistant** (relativement à Σ) ssi il n'existe pas deux termes t et t' de $\mathcal{T}(\Sigma\mathscr{C}ons)$ tels que $t \overset{*}{\leftrightarrow}_{\mathcal{R}} t'$.

Définition 2.57. Soit $\mathcal{R}$ un système de réécriture sur une signature Σ. On dit que $\mathcal{R}$ est **suffisamment complet** (relativement à Σ) ssi pour tout terme $t \in \mathcal{T}(\Sigma)$, il existe un terme $t' \in \mathcal{T}(\Sigma\mathscr{C}ons)$ tel que $t \overset{*}{\rightarrow}_{\mathcal{R}} t'$.

Remarque. Un système vérifiant les propriétés de consistance et de complétude suffisante induit une relation d'équivalence dont chaque classe contient exactement un terme de $\mathcal{T}(\Sigma\mathscr{C}ons)$.

La validité de certaines propriétés (sous forme d'égalités) dans l'algèbre $\mathcal{T}(\Sigma)/=_{\mathcal{R}}$ est démontrable par induction : il s'agit des théorèmes inductifs [Comon, 2001]. Certains outils ([Bouhoula and Rusinowitch, 1995, Stratulat, 2011]) permettent de prouver de tels théorèmes lorsque l'on dispose d'une présentation consistante et suffisamment complète d'un système de réécriture.

2.4 Automates finis d'arbres

2.4.1 Langages d'arbres

Le rôle fondamental des automates (de mots) dans la théorie des langages n'est plus à présenter. Ils permettent de décrire un moyen opérationnel pour évaluer l'appartenance d'un mot à un langage régulier et sont munis d'un certain nombre d'algorithmes réalisant les opérations pour lesquelles la classe des langages réguliers est close. La notion d'automate d'arbres est l'extension « naturelle » de celle d'automate

de mots. La relation que les automates d'arbres entretiennent avec les systèmes de réécriture de termes et la logique leur confère un intérêt particulier justifiant leur présence dans ce manuscrit.

Plusieurs présentations peuvent être données des automates d'arbres. En particulier, ils peuvent être présentés indépendamment de toute notion de systèmes de réécriture. Pour autant, nous profiterons que cette notion ait été introduite pour dresser une présentation plus succincte des automates d'arbres basée sur la réécriture de termes.

Pour une lecture plus complète sur les automates d'arbres, le lecteur est invité à consulter [Comon et al., 2008].

On rappelle qu'un langage de termes (sous-entendu sur Σ) ou langage d'arbres est une partie de $\mathcal{T}(\Sigma)$.

Définition 2.58. Soient $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature et Q un ensemble, disjoint de $\mathcal{F}$, de constantes sortées par des éléments de $\mathcal{S}$. On note $\Sigma[Q]$ la signature $(\mathcal{S}, \mathcal{F} \cup Q)$. Un élément de $\mathcal{T}(\Sigma[Q])$ est appelé **configuration** sur Σ (sous-entendu relativement à Q). Une configuration c est dite **normalisée** lorsque $c(p) \in \mathcal{F}$ seulement si $p = \varepsilon$.

Définition 2.59. On appelle **automate (fini) d'arbres** tout quadruplet $\mathbb{A} = (\Sigma, Q, F, \Delta)$ tel que :

- $\Sigma = (\mathcal{S}, \mathcal{F})$ est une signature,
- Q est un ensemble disjoint de $\mathcal{F}$ de constantes (sortées par des éléments de $\mathcal{S}$) appelés états,
- F est un sous-ensemble de Q dont les éléments sont appelés états finaux,
- Δ est un système de réécriture sur $\Sigma[Q]$ normalisé à gauche et dont les membres droits sont des états. Les règles de Δ seront appelées règles de transition de $\mathbb{A}$.

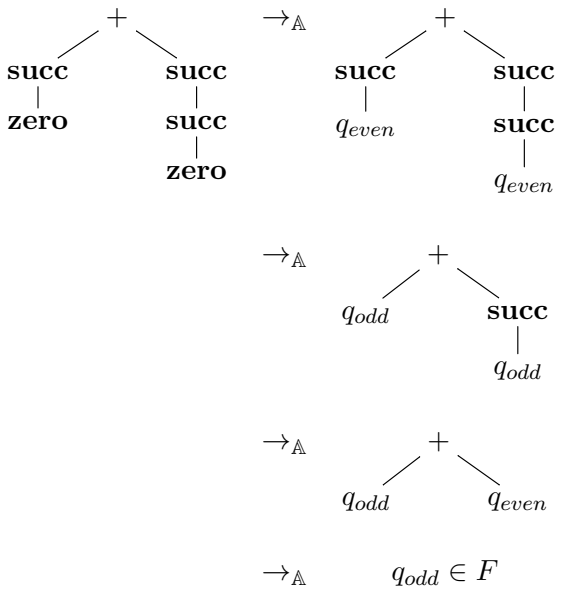
La relation de réécriture sur $\mathcal{T}(\Sigma[Q])$ induite par Δ est notée $\rightarrow_A$. On appelle **calcul** toute séquence de réductions de $\rightarrow_A$ dont la source est un terme de $\mathcal{T}(\Sigma)$.

Définition 2.60. À tout automate d'arbres $\mathbb{A} = (\Sigma, Q, F, \Delta)$ on associe un langage de termes noté $\mathcal{L}(\mathbb{A})$ et défini par : $\mathcal{L}(\mathbb{A}) = \{t \in \mathcal{T}(\Sigma) \mid \exists q_F \in F,\ t \xrightarrow{*}_{\mathbb{A}} q_F\}$. $\mathcal{L}(\mathbb{A})$ est appelé **langage reconnu** ou **engendré** par $\mathbb{A}$. Ses éléments sont dits reconnus par $\mathbb{A}$.

Exemple 2.61. Soit Σ la signature contenant les symboles $+ : \mathsf{Nat} \times \mathsf{Nat} \rightarrow \mathsf{Nat}$, $\mathbf{zero} :\rightarrow \mathsf{Nat}$ et $\mathbf{succ} : \mathsf{Nat} \rightarrow \mathsf{Nat}$. On définit l'automate reconnaissant les entiers impairs $\mathbb{A} = (\Sigma, \{q_{odd}, q_{even}\}, \{q_{odd}\}, \Delta)$ où Δ contient les règles :

$$\begin{cases} \mathbf{zero} \rightarrow q_{even} \\ \mathbf{succ}(q_{even}) \rightarrow q_{odd} \\ \mathbf{succ}(q_{odd}) \rightarrow q_{even} \\ +(q_{odd}, q_{odd}) \rightarrow q_{even} \\ +(q_{even}, q_{even}) \rightarrow q_{even} \\ +(q_{odd}, q_{even}) \rightarrow q_{odd} \\ +(q_{even}, q_{odd}) \rightarrow q_{odd} \end{cases}$$

Le terme **succ**(**zero**)+**succ**(**succ**(**zero**)) appartient au langage reconnu par $\mathbb{A}$. Pour le vérifier, il suffit d'exhiber un calcul partant de ce terme et la destination est un état de F :



Définition 2.62. Un automate d'arbre est dit **déterministe** si toute configuration normalisée est le membre gauche d'au plus une règle de transition. Il est dit **complet** si toute configuration normalisée est le membre gauche d'au moins une règle de transition.

Remarque. Comme à tout automate peut être associé un automate équivalent (c'est-à-dire reconnaissant le même langage) déterministe et complet, on peut supposer sans restriction que tous les automates considérés sont déterministes et complets.

Définition 2.63. Un langage reconnu par un automate d'arbres est dit **régulier**.

Dans la suite, toute propriété ou opération énoncée comme étant **effective** sur la classe des langages réguliers signifie que l'on dispose d'algorithmes vérifiant cette propriété ou réalisant cette opération sur les automates d'arbres correspondants.

Théorème 2.64. La classe des langages réguliers de termes est effectivement close pour les opérations ensemblistes (booléennes).

Remarque. On utilisera les notations ensemblistes $\cap$, $\cup$ et $^-$ sur les automates bien que ces derniers s'appliquent *stricto-sensu* sur les langages qu'ils reconnaissent. De manière générale, on confondra les automates avec les langages correspondants.

Il est important de noter que quelques ensembles remarquables sont réguliers :

Proposition 2.65. Soit Σ une signature et s une sorte de Σ. L'ensemble $\mathcal{T}(\Sigma)_s$ est effectivement régulier. En conséquence, l'ensemble $\mathcal{T}(\Sigma)$ l'est également.

Proposition 2.66. Soit Σ une signature et t un terme linéaire sur Σ. L'ensemble des instances closes de t, $\{\sigma(t) \mid \sigma : \mathcal{V}ar(t) \to \mathcal{T}(\Sigma)\}$ est effectivement régulier. On note $\mathcal{R}ec(t)$ l'automate correspondant.

Enfin, il est intéressant de mentionner les deux propriétés de clôture suivantes.

Proposition 2.67. Soit E un ensemble régulier et C un contexte. Les ensembles $\{C[t] \mid t \in E\}$ et $\{t \mid C[t] \in E\}$ sont effectivement réguliers.

2.4.2 Langages de tuples

Dans la suite, on parlera indifféremment de langages de tuples ou de relations de termes.

Nous distinguerons dans ce manuscrit deux classes de langages de tuples de termes : la classe des langages fortement reconnaissables et celle des langages reconnaissables. Une autre classe de langages de couples est considérée dans la littérature (celle associée à la notion de GTT (ground tree transducer)) mais ne sera pas traitée ici. Les reconnaisseurs associés à ces deux classes de langages de tuples sont basés sur la notion d'automates d'arbres présentés dans la section précédente mais se formulent sur une autre signature que celle des termes reconnus. Plus précisément, le langage de tuples reconnu par les automates que nous allons présenter sont des interprétations du langage de termes réellement reconnu.

Définition 2.68. Soit Σ une signature. On note $\Sigma^n_\#$ la signature obtenue en ajoutant à Σ, pour tout n-uplet de sorte $(\mathsf{s_1}, \ldots, \mathsf{s_n}) \in \mathcal{S}^n$,

- la sorte $[\mathsf{s_1}, \ldots, \mathsf{s_n}]$ et
- un symbole fonctionnel noté $\#^{\mathsf{s_1},\ldots,\mathsf{s_n}} : \mathsf{s_1} \times \ldots \times \mathsf{s_n} \to [\mathsf{s_1}, \ldots, \mathsf{s_n}]$.

Pour simplifier les notations, tous les symboles $\#^{s_1,\ldots,s_n}$ seront représentés par le symbole $\#$.

Définition 2.69. Soit Σ une signature. À tout automate d'arbres $\mathbb{A} = (\Sigma^n_\#, Q, F, \Delta)$ on associe le langage de tuples :

$$\mathcal{L}^\#(\mathbb{A}) = \{(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma)_{\mathsf{s_1}} \times \ldots \times \mathcal{T}(\Sigma)_{\mathsf{s_n}} \mid \#(t_1, \ldots, t_n) \xrightarrow{*}_{\mathbb{A}} q_F \in F\}$$

Un langage de tuples L est dit **fortement reconnaissable** ou **fortement régulier** ssi il existe un automate $\mathbb{A}$ tel que $L = \mathcal{L}^\#(\mathbb{A})$.

Nous allons maintenant présenter une autre notion de reconnaissabilité de tuples de termes due à Dauchet et Tison [Dauchet and Tison, 1990] et fondée sur une représentation des termes par superposition des symboles.

Définition 2.70. Soit $\Sigma = (\mathcal{S}, \mathcal{F})$ une signature. On note Σ^n la signature contenant :

- la sorte $[\mathsf{s_1}, \ldots, \mathsf{s_n}]$ pour tout n-uplet $(\mathsf{s_1}, \ldots, \mathsf{s_n}) \in (\mathcal{S} \cup \{\lambda\})^n$,
- le symbole $\langle\, \mathbf{f_1}, \ldots, \mathbf{f_n} \,\rangle$ de $(\mathcal{F} \cup \{\Lambda\})^n \setminus \{\Lambda\}^n$ dont le profil est :

$$\mathsf{S_1} \times \ldots \times \mathsf{S_k} \to [\mathsf{s_1}, \ldots, \mathsf{s_n}]$$

où pour tout $1 \leq j \leq k$, $\mathsf{S_j} = [\mathsf{s}(1,j), \ldots, \mathsf{s}(n,j)]$ où

$$\mathsf{s}(i,j) = \begin{cases} \mathsf{s_{i_j}} & si\ \mathbf{f_i} : \mathsf{s_{i_1}} \times \ldots \times \mathsf{s_{i_m}} \to \mathsf{s_i} \in \mathcal{F}\ et\ j \leq m \\ \lambda & sinon \end{cases}$$

et $k = \max_{i \in [1,n]}(ar(\mathbf{f_i}) \mid \mathbf{f_i} \neq \Lambda)$.

Puisque l'on dispose d'une signature pour représenter les tuples de termes, il nous faut nous doter d'une fonction calculant pour tout tuple de termes sa représentation dans cette nouvelle signature.

Définition 2.71. Pour tout n-uplet de termes de Σ, on note $t_1 \otimes \ldots \otimes t_n$ le terme t de Σ^n tel que pour toute position $\omega \in \bigcup_{i=1}^{n} \mathcal{P}os(t_i)$, $t(\omega) = \langle\, t_1[\omega), \ldots, t_n[\omega) \,\rangle$ où $u[\omega) = u(\omega)$ si $\omega \in \mathcal{P}os(u)$ et Λ sinon.

On peut donc enfin définir la nouvelle notion de reconnaissabilité.

Définition 2.72. Soit Σ une signature. À tout automate d'arbres $\mathbb{A} = (\Sigma^n, Q, F, \Delta)$ on associe le langage de tuples :

$$\mathcal{L}^{\otimes}(\mathbb{A}) = \{(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma)_{s_1} \times \ldots \times \mathcal{T}(\Sigma)_{s_n} \mid t_1 \otimes \ldots \otimes t_n \xrightarrow{*}_{\mathbb{A}} q_F \in F\}$$

Un langage de tuple L est dit **reconnaissable** ou **régulier** ssi il existe un automate $\mathbb{A}$ tel que $L = \mathcal{L}^{\otimes}(\mathbb{A})$.

Exemple 2.73. Si l'on considère de nouveau la signature donnée dans l'exemple précédent, on peut définir un automate reconnaissant les couples d'entiers $\langle\, n_1, n_2 \,\rangle$ tels que $n_1 \leq n_2$ (si l'on interprète les symboles **zero** et **succ** par respectivement 0 et $n \mapsto n+1$) $\mathbb{A}_{inf} = (\Sigma^2, \{q, q_{inf}\}, \{q_{inf}\}, \Delta)$ avec Δ défini par :

$$\begin{cases} \langle\, \mathbf{succ}, \mathbf{succ} \,\rangle\, (q_{inf}) & \to & q_{inf} \\ \langle\, \mathbf{zero}, \mathbf{succ} \,\rangle\, (q) & \to & q_{inf} \\ \langle\, \Lambda, \mathbf{succ} \,\rangle\, (q) & \to & q \\ \langle\, \Lambda, \mathbf{zero} \,\rangle & \to & q \end{cases}$$

Pour montrer que $\langle\, \mathbf{succ}(\mathbf{zero}), \mathbf{succ}(\mathbf{succ}(\mathbf{zero})) \,\rangle$ est reconnu par $\mathbb{A}_{inf}$, il suffit d'exhiber le calcul suivant, partant de $\mathbf{succ}(\mathbf{zero}) \otimes \mathbf{succ}(\mathbf{succ}(\mathbf{zero}) =$

$\langle \mathbf{succ}, \mathbf{succ} \rangle (\langle \mathbf{zero}, \mathbf{succ} \rangle (\langle \mathbf{\Lambda}, \mathbf{zero} \rangle)) :$

$$
\begin{array}{lll}
\begin{array}{c}\langle \mathbf{succ}, \mathbf{succ} \rangle \\ | \\ \langle \mathbf{zero}, \mathbf{succ} \rangle \\ | \\ \langle \Lambda, \mathbf{zero} \rangle\end{array} & \rightarrow_{\mathbb{A}_{inf}} & \begin{array}{c}\langle \mathbf{succ}, \mathbf{succ} \rangle \\ | \\ \langle \mathbf{zero}, \mathbf{succ} \rangle \\ | \\ q\end{array} \\
 & \rightarrow_{\mathbb{A}_{inf}} & \begin{array}{c}\langle \mathbf{succ}, \mathbf{succ} \rangle \\ | \\ q_{inf}\end{array} \\
 & \rightarrow_{\mathbb{A}_{inf}} & q_{inf}
\end{array}
$$

La relation entretenue par les deux classes de langages présentées est la suivante.

Proposition 2.74. Tout langage fortement régulier est régulier. La réciproque est fausse.

Terminons cette présentation des automates d'arbres par les propriétés qui motivent leur utilisation. Pour cela, rappelons les définitions des opérations de projection et de cylindrification.

Définition 2.75. Soit $\mathcal{R}$ une relation de termes n-aire. On appelle $i^{\text{ème}}$ projection de $\mathcal{R}$ et l'on note $\sqcap_i(\mathcal{R})$ la relation définie par :

$$(t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n) \in \sqcap_i(\mathcal{R}) \;\Leftrightarrow\; \exists t \in \mathcal{T}(\Sigma),\ (t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n) \in \mathcal{R}$$

On appelle $i^{\text{ème}}$ cylindrification de $\mathcal{R}$ et l'on note $\sqcup_i(\mathcal{R})$ la relation définie par :

$$(t_1, \ldots, t_{i-1}, t, t_i, \ldots, t_n) \in \sqcup_i(\mathcal{R}) \quad \Leftrightarrow \quad (t_1, \ldots, t_{i-1}, t_i, \ldots, t_n) \in \mathcal{R}$$

Théorème 2.76. La classe des langages de tuples fortement réguliers est effectivement close pour les opérations ensemblistes (booléennes), la projection, le produit cartésien, la cylindrification et la composition.

Théorème 2.77. La classe des langages de tuples réguliers est effectivement close pour les opérations ensemblistes (booléennes), la projection, le produit cartésien, la cylindrification et la composition.

Théorème 2.78. [Nieslon et al., 2002] La classe des langages de couples (relations binaires) fortement réguliers est effectivement close par itération (fermeture transitive reflexive).

Ce n'est en revanche pas le cas de la classe des relations régulières. Par ailleurs, il est intéressant de mentionner que certains ensembles remarquables sont fortement réguliers et d'autres seulement réguliers.

Proposition 2.79. Étant données une signature Σ et n sortes $\mathsf{s_1}, \ldots, \mathsf{s_n}$ de Σ, l'ensemble des n-uplets de termes $\{(t_1, \ldots, t_n) \mid t_1 \in \mathcal{T}(\Sigma)_{\mathsf{s_1}}, \ldots, t_n \in \mathcal{T}(\Sigma)_{\mathsf{s_n}}\}$ est fortement régulier.

En conséquence, on obtient le résultat suivant :

Proposition 2.80. L'ensemble des n-uplets de termes clos construits sur une signature donnée est fortement régulier.

En revanche, ce n'est pas le cas de la relation identité qui est seulement régulière.

Proposition 2.81. Étant donnée une signature Σ, l'ensemble des n-uplets de termes $\{(t, \ldots, t) \mid t \in \mathcal{T}(\Sigma)\}$ est régulier mais n'est pas fortement régulier.

2.5 Logique du premier ordre

Les notions de syntaxe et de sémantique sont centrales dans nos travaux. La notion d'algèbre présentée antérieurement donne un premier aperçu de la notion d'interprétation du premier ordre.

2.5.1 Syntaxe

On étend la notion de signature précédemment introduite pour intégrer de nouveaux objets appelés prédicats.

Définition 2.82. On appelle **signature logique** tout triplet $\tilde{\Sigma} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$ où :

- $\Sigma = (\mathcal{S}, \mathcal{F})$ est une signature (au sens de la définition 2.19)
- $\mathcal{P}$ est un ensemble non vide, disjoint de $\mathcal{S}$ et de $\mathcal{F}$, dont les éléments sont appelés symboles de prédicats. À tout élément $\mathbf{p} \in \mathcal{P}$ est associée une suite non vide $(\mathsf{s_1}, \ldots, \mathsf{s_n})$ d'éléments de $\mathcal{S}$ appelée **profil**. On note alors :

$$\mathbf{p} \ : \ \mathsf{s_1} \times \ldots \times \mathsf{s_n}$$

  L'entier n est appelé **arité** de $\mathbf{p}$ et est noté $ar(\mathbf{p})$.

Définition 2.83. On appelle **atome** de $\tilde{\Sigma} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$ tout objet de la forme $\mathbf{p}(t_1, \ldots, t_n)$ où $\mathbf{p} : \mathsf{s_1} \times \ldots \times \mathsf{s_n} \in \mathcal{P}$ et $t_1 \in \mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s_1}}, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})_{\mathsf{s_n}}$. L'ensemble des atomes de $\tilde{\Sigma}$ est noté $\mathcal{A}t(\tilde{\Sigma}, \mathcal{X})$. L'ensemble des atomes clos (c.-à-d. sans variable), est noté $\mathcal{A}t(\tilde{\Sigma})$.

Définition 2.84. L'ensemble des **formules** d'une signature logique $\tilde{\Sigma} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$ est le plus petit ensemble noté $\mathcal{F}or(\tilde{\Sigma})$ tel que :

- $\top$ et $\bot$ sont des formules,
- tout atome de $\mathcal{A}t(\tilde{\Sigma}, \mathcal{X})$ appartient à $\mathcal{F}or(\tilde{\Sigma})$,
- si φ et ψ sont deux formules de $\tilde{\Sigma}$ et x et y deux variables : $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\forall x.\varphi$ et $\exists x.\varphi$ sont des formules.

Les symboles $\Rightarrow, \Leftrightarrow, \ldots$ sont vus comme des raccourcis d'écriture. On appellera littéral toute formule formée uniquement d'un atome ou de la négation d'un atome.

Définition 2.85. On étend l'ensemble $\mathcal{F}or(\tilde{\Sigma})$ des formules de $\tilde{\Sigma}$ à l'ensemble des **formules avec égalité** $\mathcal{F}or^{=}(\tilde{\Sigma})$ en remplaçant dans la définition précédente $\mathcal{F}or(\tilde{\Sigma})$ par $\mathcal{F}or^{=}(\tilde{\Sigma})$ et en ajoutant l'assertion suivante : toute égalité $t = t'$ de termes de même sorte appartient à $\mathcal{F}or^{=}(\tilde{\Sigma})$.

Remarque. Toute formule possède une notation arborescente unique.

Définition 2.86. On dit qu'une variable est à la **portée** d'un quantificateur si elle apparaît dans le sous-arbre dont le quantificateur est la racine. Une variable à la portée d'un quantificateur est dite **liée** (à ce quantificateur). Une variable qui n'est pas liée est **libre**. On note respectivement $\mathcal{FV}ar(\varphi)$ et $\mathcal{BV}ar(\varphi)$ les ensembles des variables libres et liées de la formule φ.

Définition 2.87. Une formule φ est dite **polie** si elle vérifie les conditions suivantes :

- il n'existe pas de variable dont une occurence soit libre et une autre soit liée : $\mathcal{BV}ar(\varphi) \cap \mathcal{FV}ar(\varphi) = \varnothing$,
- deux occurences d'une même variable liée correspondent à la même occurence de quantificateur.

Remarque. À toute formule correspond au moins une formule polie équivalente obtenue par renommage des variables.

On supposera dans la suite que toute formule est polie.

Définition 2.88. La notion de **substitution** $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})$ est étendue aux formules de la façon suivante :

- $\sigma(\top) = \top$, $\sigma(\bot) = \bot$,
- $\sigma(p(t_1, \ldots, t_n)) = p(\sigma(t_1), \ldots, \sigma(t_n))$,
- $\sigma(\varphi \oplus \psi) = \sigma(\varphi) \oplus \sigma(\psi)$ pour tout connecteur logique $\oplus$,
- $\sigma(Qx.\varphi) = Qx.\sigma_{|\mathcal{D}om(\sigma)\setminus\{x\}}(\varphi)$ pour tout quantificateur Q (où $\sigma_{|E}$ signifie σ restreint à E).

Remarque. Le dernier point de la définition précédente nous assure qu'une substitution ne modifie que les variables libres d'une formule.

Dans la suite, on notera $\varphi[x_1, \ldots, x_n]$ pour indiquer que les variables libres de φ sont $\{x_1, \ldots, x_n\}$. La notation $\varphi[t_1, \ldots, t_n]$ réfèrera alors à la formule $\sigma(\varphi)$ pour $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$.

2.5.2 Sémantique

Définition 2.89. La notion de Σ-algèbre est étendue aux signatures logiques en définissant pour tout $\mathbf{p} : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \in \mathcal{P}$, une relation $\mathbf{p}_{\mathcal{A}} \subseteq |\mathcal{A}|_{\mathsf{s}_1} \times \ldots \times |\mathcal{A}|_{\mathsf{s}_n}$ appelée **interprétation de p**. $\mathcal{A}$ est alors appelée interprétation de $\tilde{\Sigma}$ ou $\tilde{\Sigma}$-interprétation.

Remarque. Une $\tilde{\Sigma}$-interprétation dont le domaine est $\mathcal{T}(\Sigma)$ est qualifiée d'interprétation de Herbrand.
Remarque. Toutes les notions définies dans le cadre des algèbres sont naturellement étendues à celle d'interprétations de signature logique.

Dans la suite, on note $\mathbb{B}$ l'algèbre de Boole munie de l'opération unaire $\sim$ (non) ainsi que des opérations binaires usuelles $+$ (ou) et $\times$ (et) respectivement généraliseés en $\sum$ et $\prod$.

Jacques Herbrand

Jacques Herbrand (12 février 1908 – 27 juillet 1931) était un mathématicien français né à Paris et décédé dans un accident de montagne à La Bérarde (Isère). Bien que décédé au jeune âge de 23 ans, il était déjà considéré comme l'un des plus grands mathématiciens de la jeune génération par ses professeurs Helmut Hasse et Richard Courant. Il a travaillé sur la logique mathématique ainsi que sur la théorie des corps de classes. Il a notamment introduit la notion de fonction récursive.

Définition 2.90. Soient $\tilde{\Sigma}$ une signature logique, $\mathcal{A}$ une $\tilde{\Sigma}$-interprétation et δ une $\mathcal{A}$-valuation. On étend la notion de **sémantique** dans $\mathcal{A}$ selon δ à une application $[\![.]\!]^{\delta}_{\mathcal{A}}$ de $\mathcal{F}or(\tilde{\Sigma})$ vers $\mathbb{B}$ de la façon suivante :

- $[\![\top]\!]^{\delta}_{\mathcal{A}} = 1$
- $[\![\bot]\!]^{\delta}_{\mathcal{A}} = 0$
- $[\![\mathbf{p}(t_1, \ldots, t_n)]\!]^{\delta}_{\mathcal{A}} = 1$ si $\left([\![t_1]\!]^{\delta}_{\mathcal{A}}, \ldots, [\![t_n]\!]^{\delta}_{\mathcal{A}}\right)$ appartient à $\mathbf{p}_{\mathcal{A}}$, 0 sinon,
- $[\![\neg\varphi]\!]^{\delta}_{\mathcal{A}} = \sim [\![\varphi]\!]^{\delta}_{\mathcal{A}}$
- $[\![\varphi \wedge \psi]\!]^{\delta}_{\mathcal{A}} = [\![\varphi]\!]^{\delta}_{\mathcal{A}} \times [\![\psi]\!]^{\delta}_{\mathcal{A}}$
- $[\![\varphi \vee \psi]\!]^{\delta}_{\mathcal{A}} = [\![\varphi]\!]^{\delta}_{\mathcal{A}} + [\![\psi]\!]^{\delta}_{\mathcal{A}}$
- $[\![\exists x.\varphi]\!]^{\delta}_{\mathcal{A}} = \sum_{\delta' \ \mathcal{A}\text{-valuation}} [\![\varphi]\!]^{\delta\cup\delta'}_{\mathcal{A}}$
- $[\![\forall x.\varphi]\!]^{\delta}_{\mathcal{A}} = \prod_{\delta' \ \mathcal{A}\text{-valuation}} [\![\varphi]\!]^{\delta\cup\delta'}_{\mathcal{A}}$

Définition 2.91. Soient $\tilde{\Sigma}$ une signature logique, $\mathcal{A}$ une $\tilde{\Sigma}$-interprétation et φ une formule sur $\tilde{\Sigma}$. On dit que :

- φ est **satisfiable** dans $\mathcal{A}$ s'il existe une valuation δ telle que $[\![\varphi]\!]^{\delta}_{\mathcal{A}} = 1$ (φ est dite satisfiable s'il existe une algèbre dans laquelle φ est satisfiable) ;
- φ est **valide** dans $\mathcal{A}$ si pour toute valuation δ, $[\![\varphi]\!]^{\delta}_{\mathcal{A}} = 1$, dans ce cas on dit que $\mathcal{A}$ est un modèle de φ et l'on note $\mathcal{A} \models \varphi$. On note $\mathcal{M}od(\varphi)$ l'ensemble des modèles de φ.

Remarque. La relation $\models$ est étendue aux ensembles de formules :

- une interprétation $\mathcal{A}$ est modèle d'un ensemble de formules (ou théorie) $\mathcal{T}h$ si c'est un modèle de toutes les formules de l'ensemble,
- φ est une conséquence logique de $\mathcal{T}h$ si $\mathcal{M}od(\mathcal{T}h) \subseteq \mathcal{M}od(\varphi)$, dans ce cas, on note $\mathcal{T}h \models \varphi$.

2.5.3 Sémantiques syntaxiques et constructeurs

Refermons ce chapitre en évoquant le lien entretenu entre les notions de sémantique et de symboles constructeurs. Considérons de nouveau une signature (non une signature logique) $\Sigma = (\mathcal{S}, \mathcal{F})$. Interpréter la signature Σ, c'est définir une Σ-algèbre $\mathcal{A}$ interprétant chaque terme de $\mathcal{T}(\Sigma)$ par un élément de $|\mathcal{A}|$. La plupart du temps, pour pouvoir raisonner de façon automatique, le domaine des algèbres manipulées est lui-même un ensemble de termes clos construit sur une signature donnée Σ'. On parle alors d'interprétation syntaxique. Nous évoquions un peu plus tôt la notion d'interprétation de Herbrand dans laquelle chaque terme est interprété de façon triviale par lui-même. Nous étendons ici cette notion en partitionnant l'ensemble des symboles fonctionnels en deux ensembles : l'ensemble des symboles constructeurs et celui des symboles dits définis. L'objectif est le suivant. Lorsque l'on définit une signature, il existe une partie des symboles dont la mission est de décrire les éléments du système que l'on modélise. On ne souhaite en général pas interpréter ces symboles par d'autres symboles. C'est le cas par exemple des symboles **zero** $:\rightarrow \mathsf{Nat}$ et **succ** $: \mathsf{Nat} \rightarrow \mathsf{Nat}$ modélisant les entiers naturels. Une autre partie de la signature est quant à elle destinée à être interprétée par des fonctions sur les éléments du système. C'est le cas par exemple du symbole $+ : \mathsf{Nat} \times \mathsf{Nat} \rightarrow \mathsf{Nat}$. En somme, les symboles constructeurs construisent le domaine du discours (domaine d'interprétation) et les symboles définis les fonctions. Lorsque une signature sera établie en vue d'être interprétée de cette façon, nous noterons $\mathcal{F}\mathscr{C}ons$ l'ensemble des symboles constructeurs et $\mathcal{F}\mathscr{D}ef$ l'ensemble des symboles définis. $\Sigma\mathscr{C}ons$ réfère alors à la signature constituée uniquement des symboles constructeurs et les éléments de $\mathcal{T}(\Sigma\mathscr{C}ons)$ seront appelés termes-données. Étant donnée une signature $\tilde{\Sigma}$, on appellera **interprétation syntaxique basée sur les constructeurs** toute interprétation $\mathcal{A}$ de domaine $\mathcal{T}(\Sigma\mathscr{C}ons)$ telle que tout terme-donnée est interprété par lui-même.

Au regard de ces remarques, retournons sur les notions de systèmes de réécriture et d'automates d'arbres. Un système de réécriture sur une signature Σ, convergent sur les termes clos et dont l'ensemble des formes normales est $\mathcal{T}(\Sigma\mathscr{C}ons)$, définit exactement une interprétation syntaxique de Σ. Cette interprétation interprète les symboles constructeurs de façon triviale et les symboles définis comme des fonctions sur l'ensemble des termes constructeurs. De la même façon, les automates d'arbres définissent des interprétations syntaxiques de prédicats dès lors qu'ils sont construits sur la signature des symboles constructeurs.

3 Spécification, analyse et transformation de politiques de sécurité

Sommaire

3.1 Spécification des modèles et des politiques 50
3.1.1 Vocabulaire 50
3.1.2 Modèles de sécurité 51
3.1.3 Politiques de sécurité 54
3.2 Sémantique et propriétés 57
3.3 Changement de modèle 60
3.3.1 Positionnement du problème 60
3.3.2 Méthode 62
3.4 Travaux reliés 67
3.5 Synthèse 69

Lorsque l'on prend la décision de créer un logiciel, l'une des principales questions qui se posent au développeur est le choix du langage de programmation qu'il va utiliser. Son choix va être établi en fonction de l'architecture globale du projet dans lequel le logiciel s'inscrit, du paradigme de programmation qui lui permettra de formuler le plus aisément possible les structures et les opérations nécessaires mais également en fonction de sa propre sensibilité. Le concepteur d'une politique d'autorisations devra faire face aux mêmes interrogations. Au delà du langage, le choix du modèle de sécurité sur lequel il va fonder sa gestion des autorisations est déterminante pour la bonne administration de cette dernière. Nous l'avons évoqué, de multiples modèles ont été développés depuis les années 70 pour faciliter la formulation d'une

politique en fonction du contexte dans lequel elle s'inscrit [Barker, 2009]. De plus, les systèmes d'information sont rarement immuables. Ils changent au gré de l'évolution de l'organisation qu'ils supportent (fusion, réorganisation, ...) ou pour améliorer la gestion de l'information. Dans ce contexte profondément dynamique, maintenir une politique de sécurité peut engendrer la nécessité de changer le modèle dans lequel la politique se définit. La migration d'une politique d'un modèle vers un autre est une tâche difficile et critique. Lorsqu'elle est faite manuellement, elle est sujette à l'introduction d'erreur et peut engendrer des fuites de sécurité. C'est la raison pour laquelle, dans l'objectif de préserver la confiance que l'on porte en une politique de sécurité après sa migration, nous proposons que la traduction de la politique se fasse d'une façon automatique et basée sur des algorithmes formellement vérifiés. Dans ce chapitre, nous nous intéressons à l'établissement d'un cadre de spécification fondé sur une sous-classe de la logique du premier ordre ainsi que sur les techniques liées aux automates d'arbres. Plus précisément, nous proposons la spécification de politiques en deux étapes disctinctes. La première concerne la formalisation du modèle de sécurité sous la forme de règles associant les actions à des contraintes du premier ordre. La seconde consiste, quant à elle, à permettre la configuration des éléments de sécurité à l'aide de programmes logiques. Nous nous attachons à montrer que cette approche préserve les avantages des spécifications monoblocs et qu'elle permet d'investiguer de nouvelles possibilités. En particulier, nous montrons que de telles spécifications sont opérationnelles (en ce sens où l'on en déduit un processus de calcul des autorisations) et qu'elles peuvent être interrogées (dans le sens des requêtes dites administratives). Nous proposons également un algorithme qui transforme une politique exprimée dans un modèle vers une politique équivalente exprimée dans un autre modèle.

3.1 Spécification des modèles et des politiques

3.1.1 Vocabulaire

Nous allons naturellement débuter ce chapitre en exposant le langage de spécification des modèles et politiques de sécurité. Mais avant cela, convenons d'une mise au point sur le vocabulaire emprunté. Comme nous l'avons indiqué, nous préconisons, dans un objectif de réutilisabilité, la spécification des politiques en deux phases. La partie la plus générique est appelée modèle de sécurité et correspond à la définition que nous en avons donné dans le chapitre précédent : le modèle indique les éléments de sécurité à prendre en compte, les relations entretenues par ces éléments ainsi que le schéma de calcul (ou de déduction) des autorisations d'action. Par exemple, pour spécifier un modèle basé sur les niveaux de sécurité, on peut indiquer que chaque sujet et objet doit être affecté à un niveau de sécurité et que la lecture n'est possible que lorsqu'un sujet possède un niveau de sécurité supérieur à celui de l'objet. En revanche, la spécification du modèle ne donne aucune information sur les niveaux de sécurité en question. C'est l'objet de ce que l'on appelle dans ce chapitre la configuration de la politique. La configuration consiste en la définition de l'évaluation des fonctions et des prédicats définis par le modèle de sécurité. Pour reprendre l'exemple d'une politique basée sur les niveaux de sécurité, l'objectif de la

configuration est d'assigner les niveaux aux sujets et objets. Pour définir le vocabulaire décrivant les différentes données appartenant au système d'information (sujets, objets, actions, ...), nous introduisons la notion de signature de sécurité.

Définition 3.1. Une **signature de sécurité** est une signature $\tilde{\Sigma} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$ dont les symboles de prédicats sont appelés symboles actions.

La signature [1] $\tilde{\Sigma}$ décrit les éléments de sécurité appartenant au système qui ne sont pas spécifiques à la politique qui sera définie. Il s'agit par exemple des utilisateurs, des fichiers mais ne spécifie pas, par exemple, la notion de rôle qui est spécifique aux politiques basées sur les rôles. Les atomes construits sur $\tilde{\Sigma}$ sont les actions du système. Les termes construits sur Σ sont appelés termes de données et deux termes distincts dans $\mathcal{T}(\Sigma)$ représentent des éléments différents du système d'information. La signature Σ est donc uniquement constituée de symboles constructeurs.

Exemple 3.2. Considérons un système contenant trois utilisateurs (par exemple Alice, Bob et Charlie) ainsi qu'un ensemble dénombrable de fichiers identifiés par les entiers naturels. Considérons également que les actions possibles dans le systèmes sont la lecture et l'écriture d'un fichier par un utilisateur. Une signature de sécurité correspondante à ce système est $\tilde{\Sigma} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$ où $\mathcal{S}$ contient les sortes $\mathsf{Subject}$, Object, Nat, $\mathcal{F}$ contient les symboles suivants :

$$\begin{cases} \mathbf{Alice}, \mathbf{Bob}, \mathbf{Charlie} & : & \to \mathsf{Subject} \\ \mathbf{file} & : \mathsf{Nat} & \to \mathsf{Object} \\ \mathbf{zero} & : & \to \mathsf{Nat} \\ \mathbf{succ} & : \mathsf{Nat} & \to \mathsf{Nat} \end{cases}$$

et $\mathcal{P}$ est composé des symboles d'actions **read** et **write** de profil $\mathsf{Subject} \times \mathsf{Object}$. L'atome $\mathbf{read}(\mathbf{Alice}, \mathbf{file}(\mathbf{succ}(\mathbf{zero})))$ décrit l'action correspondant à la lecture du fichier identifié par le nombre 1 (symbolisé par $\mathbf{succ}(\mathbf{zero})$) par Alice.

3.1.2 Modèles de sécurité

Comme nous l'avons déjà indiqué, les modèles de sécurité spécifient les mécanismes à implanter dans le système pour garantir les propriétés de sécurité désirées (confidentialité, intégrité, ...). Dans le cadre proposé, le modèle de sécurité décrit les structures qui lui sont spécifiques, les règles permettant de calculer les autorisations ainsi que la forme des configurations « acceptables ». La définition suivante précède une explication des différents choix qui sont exprimés dans cette dernière.

Définition 3.3. Un **modèle de sécurité** $\mathfrak{mod}$ sur une signature de sécurité $\tilde{\Sigma} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$ est la donnée de :

1. On rappelle (définition 2.82) que la notation Σ est utilisée pour les signatures « fonctionnelles » (c.-à-d. sans prédicat) tandis que $\tilde{\Sigma}$ dénote une signature « logique » (c.-à-d. avec symboles de prédicats). Lorsqu'une signature logique $\tilde{\Sigma} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$ est définie, alors Σ réfère à $(\mathcal{S}, \mathcal{F})$.

- une extension de Σ notée $\tilde{\Sigma}_{\mathtt{mod}} = (\mathcal{S} \cup \mathcal{S}_{\mathtt{mod}}, \mathcal{F} \cup \mathcal{F}_{\mathtt{mod}}, \mathcal{P}_{\mathtt{mod}})$ telle que :
  1. pour tout symbole $\mathbf{f} : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \to \mathsf{s} \in \mathcal{FC}ons_{\mathtt{mod}}$, $\mathsf{s} \in \mathcal{S}_{\mathtt{mod}}$ (c.-à-d. on ne crée pas de nouveaux constructeurs de la signature de sécurité),
  2. pour tout symbole $\mathbf{f} : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \to \mathsf{s} \in \mathcal{FD}ef_{\mathtt{mod}}$, s est finiment constructible, c.-à-d. $\mathcal{T}(\Sigma\mathcal{C}ons)_{\mathsf{s}}$ est fini,
- un ensemble de règles de sécurité $\Gamma_{\mathtt{mod}}$ de la forme :

$$action \mapsto \varphi$$

  où *action* est un atome linéaire de $\mathcal{A}t(\tilde{\Sigma}, \mathcal{X})$ appelé motif d'action et φ est une formule de $\mathcal{F}or^{=}(\tilde{\Sigma}_{\mathtt{mod}})$ telle que pour toute égalité $t = t'$, t et t' sont d'une sorte finiment constructible,
- une théorie du premier ordre (c.-à-d. un ensemble de formules) $\mathscr{T}_{\mathtt{mod}} \subseteq \mathcal{F}or^{=}(\tilde{\Sigma}_{\mathtt{mod}})$ dans laquelle pour toute égalité $t = t'$, t et t' sont d'une sorte finiment constructible et
- un sous-ensemble de sortes $\mathcal{S}^{+}_{\mathtt{mod}}$ de $\mathcal{S}_{\mathtt{mod}}$.

Le choix de ces différents éléments se justifie de la manière suivante :

- L'extension de la signature de sécurité décrit les nouvelles informations propres au modèle de sécurité. Par exemple, un modèle basé sur des niveaux de sécurité peut définir la nouvelle sorte Level ainsi qu'un ordre sur les niveaux caractérisé par un prédicat **inf** : Level × Level. La première condition sur cette extension indique que le modèle ne doit pas créer de nouveaux termes de donnée sortés par la signature de sécurité. Par exemple, dans le cas où la signature de sécurité comporte les sortes Subject et Object, le modèle ne peut pas définir de nouveaux sujets et objets. La seconde condition indique que les interprétations de fonctions sont à valeurs dans des ensembles finis (condition nécessaire pour garantir des résultats de décidabilité ultérieurs). C'est le cas, par exemple, de la fonction qui attribue les niveaux de sécurité aux sujets et objets si l'ensemble des niveaux est fixé par le modèle.
- Les règles de sécurité définissent le schéma de calcul des autorisations, c.-à-d. les conditions à satisfaire pour qu'une action soit permise.
- Comme expliqué précédemment, la sémantique des informations de sécurité n'est pas connue au niveau de la définition du modèle mais le modèle peut contraindre les interprétations des symboles qu'elle définit. Par exemple, dans le cas de la définition d'une relation d'ordre partielle **inf**, la théorie devra préciser que l'interprétation de **inf** doit respecter la contrainte $\forall x, y, z : \mathbf{inf}(x, y) \wedge \mathbf{inf}(y, z) \Rightarrow \mathbf{inf}(x, z)$.
- Enfin, le modèle peut spécifier l'existence de termes d'une sorte donnée sans en préciser les constructeurs. Par exemple, le modèle peut indiquer la présence de niveaux de sécurité mais ne pas les définir. Autrement dit, le modèle indique que c'est à l'étape de configuration que devront être définis les niveaux souhaités par l'administrateur. Dans ce cas, $\mathcal{S}^{+}_{\mathtt{mod}}$ doit contenir Level.

Exemple 3.4. Dans cet exemple, nous nous proposons de spécifier un exemple très simple pour illustrer la définition précédente. Nous nous basons sur une description simplifiée du modèle introduit dans [Sandhu et al., 1996] dans lequel les actions possibles sont la lecture et l'écriture. Rappelons que l'idée principale du modèle RBAC est d'assigner chaque utilisateur à un ensemble de rôles. Le modèle attribue également des modes d'accès (**read** et **write**) sur les ressources (les fichiers dans notre cas) aux rôles. Enfin, un utilisateur sera habilité à effectuer une action sur une ressource ssi il possède un rôle disposant du mode d'accès correspondant sur le fichier en question. Dans notre cadre, ceci se traduit de la façon suivante. Notons $\tilde{\Sigma}$ la signature de sécurité définie dans l'exemple précédent et faisant état d'un système composé d'utilisateurs et de fichiers. Les informations de sécurité définies par le modèle sont les rôles et les modes d'accès. De ce fait, $\Sigma_{\mathfrak{rbac}}$ étend $\tilde{\Sigma}$ avec les sortes Role et Mode. Nous considérons dans notre exemple que l'ensemble des rôles doit être défini par l'administrateur mais que les modes d'accès sont fixés par le modèle. Dans ce cadre, $\Sigma_{\mathfrak{rbac}}$ est composé des constructeurs (en l'occurence des constantes) **read_mode** et **write_mode** de sorte Mode correspondant respectivement aux modes de lecture et d'écriture tandis qu'aucun constructeur de sorte Role n'est défini et alors $\mathcal{S}^{+}_{\mathfrak{rbac}} = \{\mathsf{Role}\}$. Nous utiliserons le prédicat **ura** : $\mathsf{Subject} \times \mathsf{Role}$ pour représenter l'assignation des rôles aux utilisateurs (**ura** pour *u*ser-*r*ole *a*ssignment) et le prédicat **pra** : $\mathsf{Role} \times \mathsf{Mode} \times \mathsf{Object}$ pour l'affectation des privilèges d'accès aux rôles (**pra** pour *p*rivilege-*r*ole *a*ssignment). Les règles de sécurité se définissent simplement de la façon suivante :

$$\begin{cases} \mathbf{read}(s,o) \mapsto \exists r : \ \mathbf{ura}(s,r) \wedge \mathbf{pra}(r, \mathbf{read_mode}, o) \\ \mathbf{write}(s,o) \mapsto \exists r : \ \mathbf{ura}(s,r) \wedge \mathbf{pra}(r, \mathbf{write_mode}, o) \end{cases}$$

Dans la suite, nous noterons $\mathfrak{rbac}$ le modèle de sécurité ainsi défini.

Comme tous les exemples, ce dernier n'illustre que partiellement les différents choix exposés dans la définition 3.3. L'utilisation de symboles définis et de théories sera illustrée dans l'exemple 3.23.

Le lecteur attentif remarquera que la définition donnée des modèles de sécurité n'induit pas une façon unique de calculer les autorisations. Par exemple, quel sens donner à un modèle comportant les règles $\mathbf{read}(s,o) \mapsto \varphi_1$ et $\mathbf{read}(s,o) \mapsto \varphi_2$? Pour définir un schéma unique de calcul des autorisations, il est nécessaire (et suffisant) de munir les modèles d'une stratégie d'interprétation.

Définition 3.5. Une **stratégie d'interprétation** indique la façon dont les autorisations d'action doivent être évaluées en fonction des règles de sécurité du modèle. En d'autres termes, elle associe à chaque action une formule à vérifier dont le calcul de la validité correspond à l'évaluation de l'autorisation d'action. Nous définissons les stratégies suivantes :

- `and`, qui permet l'exécution d'une action ssi toutes les contraintes associées

à un motif d'action filtrant l'action considérée sont satisfaites :

$$\Gamma^{\texttt{and}}_{\texttt{mod}} : ac \mapsto \bigwedge_{\substack{a \mapsto \varphi \in \Gamma_{\texttt{mod}} \\ \sigma(a) = ac}} \sigma(\varphi)$$

- `or` : qui permet l'exécution d'une action ssi au moins une des contraintes associées à un motif d'action filtrant l'action considérée est satisfaite :

$$\Gamma^{\texttt{or}}_{\texttt{mod}} : ac \mapsto \bigvee_{\substack{a \mapsto \varphi \in \Gamma_{\texttt{mod}} \\ \sigma(a) = ac}} \sigma(\varphi)$$

- `or_else`$^{\nu}$, qui permet l'exécution d'une action ssi la contrainte associée au premier motif d'action filtrant l'action courante est satisfait et, dans le cas où l'action ne filtre aucun motif d'action, procède à la décision par défaut (autorisation dans le cas $\nu = \top$ et refus dans le cas $\nu = \bot$) :

$$\Gamma^{\texttt{or_else}^{\nu}}_{\texttt{mod}} : ac \mapsto \begin{cases} \sigma(\varphi_i) \ si \ \sigma(a_i) = ac \ et \ \forall j < i, \nexists \sigma' : \sigma'(a_j) = ac \\ \nu \ sinon \end{cases}$$

où $\nu \in \{\top, \bot\}$ et où les règles de $\Gamma_{\texttt{mod}}$ sont supposées être ordonnées.

Notons que par définition, l'autorisation par défaut (c.-à-d. lorsqu'aucun motif d'action ne correspond à l'action à évaluer) pour la stratégie `and` est $\top$ et correspond à $\bot$ pour la stratégie `or`.

Nous supposerons dans la suite que tous les modèles sont munis d'une stratégie d'interprétation qui sera indiquée en exposant du modèle, sauf lorsque toutes les stratégies coïncident (c.-à-d. lorsque chaque action possible filtre exactement un motif, ce qui est le cas dans notre précédent exemple).

3.1.3 Politiques de sécurité

Pour définir entièrement une politique de sécurité, il faut établir le modèle sur lequel la politique se fonde, les données qui lui sont spécifiques ainsi que l'interprétation de la signature du modèle. Pour définir de telles interprétations, nous nous sommes orienté vers une classe de programmes logiques d'intérêt puisqu'elle offre à la fois un pouvoir d'expression satisfaisant tout en possédant un certain nombre de propriétés qui seront essentielles par la suite. Cette classe est connue sous le nom $\mathcal{H}_1$ et a été introduite dans [Nieslon et al., 2002]. Pour définir cette classe, quelques définitions auxiliaires sont nécessaires.

Définition 3.6. Une **clause** $\mathcal{H}_1$ sur $\tilde{\Sigma}$ est une formule de la forme $H \Leftarrow \alpha$ telle que :

- H est un atome linéaire sur $\tilde{\Sigma}$ appelé tête de la clause,
- α est une conjonction d'atomes $\alpha_1 \wedge \ldots \wedge \alpha_n$ sur $\tilde{\Sigma}$ appelée corps de la clause,
- pour tout couple de variables $(x, y) \in \mathcal{V}ar(H)^2$, $x \approx_\alpha y$ seulement si $x \uparrow_H y$ où

  - $\approx_\alpha$ est la plus petite relation d'équivalence contenant $\bigcup_{i=1}^{n} \mathcal{V}ar(\alpha_i)^2$ et
  - $x \uparrow_H y$ ssi x et y sont « frères » dans H, c.-à-d. s'il existe une position ω et deux entiers k, k' tels que $H_{|\omega.k} = x$ et $H_{|\omega.k'} = y$.
- tous les symboles fonctionnels apparaissant dans H et α sont des constructeurs.

On appelle **programme** $\mathcal{H}_1$ tout ensemble de clauses $\mathcal{H}_1$.

Exemple 3.7. L'exemple qui suit a pour objectif d'aider le lecteur dans sa compréhension des restrictions syntaxiques caractérisant les clauses $\mathcal{H}_1$. Il est donc décorrélé du fil conducteur liant les autres exemples de ce chapitre. Soit $\tilde{\Sigma}$ une signature logique comportant une unique sorte (quelconque), les symboles fonctionnels $\mathbf{f}$ et $\mathbf{g}$ d'arités respectives 2 et 1 ainsi qu'un symbole de prédicat $\mathbf{p}$ d'arité 2. La clause suivante :

$$\mathbf{p}(\mathbf{f}(x_1, x_2), \mathbf{g}(\mathbf{f}(y_1, y_2))) \Leftarrow \mathbf{p}(x_1, x_2) \wedge \mathbf{p}(y_1, y_2)$$

est une clause $\mathcal{H}_1$. En effet,

- d'une part, $x_1 \approx_\alpha x_2$ (car ces deux variables apparaissent dans l'atome $\mathbf{p}(x_1, x_2)$) et $y_1 \approx_\alpha y_2$ (car ces deux variables apparaissent dans l'atome $\mathbf{p}(y_1, y_2)$),
- d'autre part, $x_1 \uparrow_H x_2$ (car $H_{|1.1} = x_1$ et $H_{|1.2} = x_2$) et $y_1 \uparrow_H y_2$ (car $H_{|2.1.1} = y_1$ et $H_{|2.1.2} = y_2$).

En revanche, si l'on ajoute l'atome $\mathbf{p}(x_1, y_1)$ au corps de la clause, alors il ne s'agit plus d'une clause $\mathcal{H}_1$. En effet, l'ajout de cet atome aggrège les deux classes d'équivalences engendrées par $\approx_\alpha$: $x_1 \approx_\alpha x_2 \approx_\alpha y_1 \approx_\alpha y_2$ alors que $\uparrow_H$ est inchangée.

La sous-classe du fragment de Horn que l'on considère possède une propriété essentielle pour la suite énoncée par le théorème suivant :

Théorème 3.8 ([Nieslon et al., 2002]). Pour tout programme P de $\mathcal{H}_1$ sur $\tilde{\Sigma} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$, la plus petite interprétation de Herbrand $\mathcal{A}$ de $\tilde{\Sigma}$ modèle de P est effectivement calculable. De plus, pour tout $\mathbf{p} \in \mathcal{P}$, $\mathbf{p}_\mathcal{A}$ est effectivement fortement régulier.

On étend la définition de la classe des programmes $\mathcal{H}_1$ en autorisant certaines égalités entre termes.

Définition 3.9. Une clause $\mathcal{H}_1^=$ sur $\tilde{\Sigma}$ est une formule de la forme $H \Leftarrow \alpha$ telle que :

- H est un atome linéaire sur $\tilde{\Sigma}$ dont les sous-termes sont des termes constructeurs ou une égalité de la forme $\mathbf{f}(t_1, \ldots, t_n) = t$ avec $\mathbf{f}$ symbole défini, t_i des termes constructeurs et t un terme de donnée,
- α est une conjonction d'atomes et d'égalités $\alpha_1 \wedge \ldots \wedge \alpha_n$ vérifiant les mêmes conditions que H,

• pour tout couple de variables $(x, y) \in \mathcal{V}ar(H)^2$, $x \approx_\alpha y$ seulement si $x \uparrow_H y$

On appelle programme $\mathcal{H}_1^=$ tout ensemble de clauses $\mathcal{H}_1^=$.

Exemple 3.10. Reprenons l'exemple 3.7 en considérant un nouveau symbole fonctionnel défini **h** ainsi que les constructeurs **zero** et **succ**. La clause suivante :

$$\mathbf{h}(\mathbf{f}(x_1, x_2), \mathbf{g}(\mathbf{f}(y_1, y_2))) = \mathbf{succ}(\mathbf{zero}) \Leftarrow \mathbf{p}(x_1, x_2) \wedge \mathbf{p}(y_1, y_2)$$

est une clause $\mathcal{H}_1^=$.

Théorème 3.11. Pour tout programme P de $\mathcal{H}_1^=$ sur $\tilde{\Sigma} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$, il est décidable de savoir s'il existe une plus petite interprétation syntaxique basée sur les constructeurs $\mathcal{A}$ de $\tilde{\Sigma}$ modèle de P. Dans ce cas, $\mathcal{A}$ est effectivement calculable et pour tout $\mathbf{p} \in \mathcal{P}$, $\mathbf{p}_\mathcal{A}$ et $=_\mathcal{A}$ sont effectivement fortement réguliers.

❝ *Démonstration.* Comme les membres droits des égalités autorisées sont des termes de donnée, on peut remplacer toute égalité $\mathbf{f}(t_1, \ldots, t_n) = t$ du programme par un atome $Eq^t_\mathbf{f}(t_1, \ldots, t_n)$ où $Eq^t_\mathbf{f}$ est un nouveau symbole de prédicat. On obtient alors un programme $\mathcal{H}_1$. Le programme initial possède un plus petit modèle syntaxique $\mathcal{A}$ basé sur les constructeurs ssi le nouveau programme possède un plus petit modèle de Herbrand $\mathcal{A}'$ tel que pour tout symbole défini **f**, les relations interprétant les différents $Eq^t_\mathbf{f}$ sont deux à deux disjointes, ce qui est décidable. ❞

Disposant d'une large classe de programmes logiques pour décrire des interprétations fortement régulières, nous proposons la définition suivante de politique de sécurité.

Définition 3.12. Une **politique de sécurité** $\mathfrak{pol}$ sur une signature de sécurité $\tilde{\Sigma}$ consiste en la donnée :

- d'un modèle de sécurité (muni d'une stratégie si nécessaire) $\mathfrak{mod}^\zeta$ sur $\tilde{\Sigma}$,
- d'un ensemble fini de constructeurs (constants) κ de sorte $s \in \mathcal{S}^+_{\mathfrak{mod}}$, étendant ainsi $\tilde{\Sigma}_{\mathfrak{mod}}$ en une autre signature notée $\tilde{\Sigma}^\kappa_{\mathfrak{mod}}$ et
- d'un programme $\mathcal{H}_1^=$ sur $\tilde{\Sigma}^\kappa_{\mathfrak{mod}}$ (éventuellement étendu avec un ensemble fini de prédicats auxiliaires) noté ρ et appelé configuration de $\mathfrak{pol}$.

La politique composée de $\mathfrak{mod}^\zeta$, κ et ρ est notée $\langle\, \mathfrak{mod}^\zeta, \rho^\kappa \,\rangle$.

Exemple 3.13. Considérons le modèle $\mathfrak{rbac}$ décrit dans l'exemple précédent et spécifions une politique sur $\mathfrak{rbac}$. Comme $\mathcal{S}^+_{\mathfrak{rbac}} = \{\mathsf{Role}\}$, il est nécessaire de définir des constructeurs de sorte Role. Soit $\kappa = \{r_1, r_2 :\rightarrow \mathsf{Role}\}$ et la

configuration donnée par le programme ρ suivant :

$$\begin{cases} \mathbf{ura}(\mathbf{Alice}, r_1) & \Leftarrow \\ \mathbf{ura}(\mathbf{Bob}, r_2) & \Leftarrow \\ \mathbf{ura}(\mathbf{Charlie}, r_1) & \Leftarrow \\ \mathbf{ura}(\mathbf{Charlie}, r_2) & \Leftarrow \\ \mathbf{pra}(r_1, \mathbf{read_mode}, \mathbf{file}(n)) & \Leftarrow \mathbf{even}(n) \\ \mathbf{pra}(r_2, \mathbf{write_mode}, \mathbf{file}(n)) & \Leftarrow \mathbf{thd}(n) \\ \mathbf{thd}(\mathbf{zero}) & \Leftarrow \\ \mathbf{thd}(\mathbf{succ}(n)) & \Leftarrow \mathbf{rem}^3_2(n) \\ \mathbf{rem}^3_2(\mathbf{succ}(n)) & \Leftarrow \mathbf{rem}^3_1(n) \\ \mathbf{rem}^3_1(\mathbf{succ}(n)) & \Leftarrow \mathbf{thd}(n) \\ \mathbf{even}(\mathbf{zero}) & \Leftarrow \\ \mathbf{odd}(\mathbf{succ}(n)) & \Leftarrow \mathbf{even}(n) \\ \mathbf{even}(\mathbf{succ}(n)) & \Leftarrow \mathbf{odd}(n) \end{cases}$$

Les prédicats auxiliaires **even**, **odd** et **thd** caractérisent respectivement les entiers pairs, impairs et multiples de trois tandis que l'atome $\mathbf{rem}^k_m(n)$ caractérise le fait que le reste de la division euclidienne de n par k est m. Cette spécification de configuration indique que Alice est affectée au rôle r_1, Bob au rôle r_2 et Charlie aux rôles r_1 et r_2. Elle caractérise également le fait que le rôle r_1 dispose du droit d'accès en lecture sur les fichiers identifiés par un entier pair et que le rôle r_2 dispose du droit d'accès en écriture sur les fichiers identifiés par un entier multiple de trois.

3.2 Sémantique et propriétés

Dans cette section, nous allons montrer que notre cadre permet de montrer des propriétés sur les spécifications de politiques. Nous montrerons également que l'on peut calculer automatiquement une procédure d'évaluation des autorisations (la sémantique de la politique). Enfin, nous montrerons que nous pouvons interroger les politiques au moyen de « requêtes administratives ». Avant cela, il est nécessaire de vérifier que la configuration de la politique est cohérente. Il est en effet nécessaire que le programme de configuration définisse une interprétation des symboles du modèle sur lequel est basé la politique et qu'il respecte les contraintes imposées par le modèle. C'est ce que nous appelons dans la suite la cohérence de la politique.

Définition 3.14. Une politique de sécurité $\langle\, \mathfrak{mod}^{\zeta}, \rho^{\kappa} \,\rangle$ est **cohérente** ssi :

(*i*) ρ possède un plus petit modèle syntaxique basé sur les constructeurs $\mathcal{A}_{min}$ de $\tilde{\Sigma}_{\mathfrak{mod}}$,

(*ii*) $\rho \models \mathscr{T}_{\mathfrak{mod}}$ (signifiant $\mathcal{A}_{min} \models \mathscr{T}_{\mathfrak{mod}}$)

La définition ainsi énoncée peut être vue comme une propriété de modélisation (propriété d'existence d'un modèle logique d'une spécification) mais peut également être vue comme la définition d'une propriété sur une politique : le premier point

correspondant à l'absence de conflit et le second correspondant à une notion de conformité de la configuration avec le modèle de sécurité.

Proposition 3.15. La cohérence d'une politique de sécurité est décidable.

Pour démontrer cette proposition ainsi que celles qui seront énoncées ultérieurement, nous nous basons sur le théorème suivant :

Théorème 3.16. Soit $\tilde{\Sigma}$ une signature logique et $\mathcal{A}$ une interprétation syntaxique basée sur les constructeurs telle que pour tout prédicat $\mathbf{p}$, $\mathbf{p}_\mathcal{A}$ est une relation fortement régulière. Pour toute formule du premier ordre φ telle que $\mathcal{FV}ar(\varphi) = \{x_1, \ldots, x_n\}$, l'ensemble $\mathcal{S}ol_\mathcal{A}(\varphi) = \{(t_1, \ldots, t_n) \mid \mathcal{A} \models \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}(\varphi)\}$ est effectivement fortement régulier.

❝ *Démonstration.* Toute relation fortement régulière R est équivalente à une union finie de produits cartésiens d'ensembles réguliers R^i_j :

$$R = \bigcup_{i=1}^{m} \left(R^i_1 \times \ldots \times R^i_n\right)$$

Notons $unary_\mathcal{A}(\varphi)$ la formule obtenue à partir de φ après remplacement de tout atome $\mathbf{p}(t_1, \ldots, t_n)$ (y compris les égalités) par $\bigvee_{i=1}^{m_{\mathbf{p}_\mathcal{A}}} \left((\mathbf{p}_\mathcal{A})^i_1(x_1) \wedge \ldots \times (\mathbf{p}_\mathcal{A})^i_n(x_n)\right)$ où les $(\mathbf{p}_\mathcal{A})^i_j$ sont issus de la décomposition de la relation $\mathbf{p}_\mathcal{A}$ en une union finie de produits cartésiens d'ensembles réguliers équivalente. Naturellement, $\mathcal{A} \models \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}(\varphi)$ ssi $\mathcal{A} \models \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}(unary_\mathcal{A}(\varphi))$. Il faut et il suffit ensuite d'appliquer les étapes suivantes :

- transformer tous les quantificateurs en quantificateurs existentiels ;
- mettre la formule sous la forme d'une disjonction de conjonctions ;
- descendre les quantificateurs le plus bas possible dans l'arbre de la formule ;
- transformer les conjonctions d'atome comportant la même variable en intersection : $\mathbf{p}_\mathcal{A}(x) \wedge \mathbf{p}'_\mathcal{A}(x)$ devient $(\mathbf{p}_\mathcal{A} \cap \mathbf{p}'_\mathcal{A})(x)$;
- remplacer les atomes niés $\neg\mathbf{p}_\mathcal{A}(x)$ par $\overline{\mathbf{p}_\mathcal{A}}(x)$ (où l'inversion de $\mathbf{p}_\mathcal{A}$ s'effectue relativement à la sorte de x, c.-à-d. : $\overline{\mathbf{p}_\mathcal{A}} = \mathcal{T}(\Sigma)_\mathsf{s} \setminus \mathbf{p}_\mathcal{A}$ où s est la sorte de x) ;
- remplacer les conjonctions contenant un atome quantifié $\exists x{:}\mathbf{p}_\mathcal{A}(x) \wedge conj$ par $conj$ si $\mathbf{p}_\mathcal{A}$ est non vide et par $\varnothing$ sinon ;
- transformer toute conjonction non vide $conj$ en $\mathcal{T}(\Sigma)(x)_\mathsf{s} \wedge conj$ pour toute variable $x : \mathsf{s} \in \mathcal{FV}ar(\varphi) \setminus \mathcal{V}ar(conj)$;
- choisir un ordre sur les variables et « ordonner les conjonctions » de telle sorte à ce que l'apparition des atomes dans chaque conjonction respecte cet ordre ;
- remplacer toutes les conjonctions qui sont alors de la forme $\mathbf{p}^1_\mathcal{A}(x_1) \wedge \ldots \wedge \mathbf{p}^n_\mathcal{A}(x_n)$ par $(\mathbf{p}^1_\mathcal{A} \times \ldots \times \mathbf{p}^n_\mathcal{A})(x_1, \ldots, x_n)$;
- remplacer la disjonction par une union : $\mathbf{p}^1_\mathcal{A}(x_1) \vee \ldots \vee \mathbf{p}^n_\mathcal{A}(x_n)$ devient $(\mathbf{p}^1_\mathcal{A} \cup \ldots \cup \mathbf{p}^n_\mathcal{A})(x_1, \ldots, x_n)$.

La formule est alors transformée en une expression caractérisant les opérations à effectuer sur les ensembles $\mathbf{p}_\mathcal{A}$, où $\mathbf{p} \in \mathcal{P}$, pour obtenir l'automate reconnaissant l'ensemble des n-uplets de valeurs validant la formule φ dans $\mathcal{A}$. ❞

Définition 3.17. Soit $\mathfrak{pol} = \langle\, \mathfrak{mod}^{\zeta}, \rho^{\kappa} \,\rangle$ une politique de sécurité cohérente. On appelle **sémantique** de $\mathfrak{pol}$ et l'on note $[\![\mathfrak{pol}]\!]$ l'ensemble des actions que $\mathfrak{pol}$ autorise, c.-à-d. :

$$[\![\mathfrak{pol}]\!] = \left\{ ac \in \mathcal{A}t(\tilde{\Sigma}) \mid \rho \models \Gamma^{\zeta}_{\mathfrak{mod}}(ac) \right\}$$

Cette définition induit une relation d'équivalence sur les politiques de sécurité : $\mathfrak{pol} \approx \mathfrak{pol}'$ ssi $[\![\mathfrak{pol}]\!] = [\![\mathfrak{pol}']\!]$.

Exemple 3.18. Soit $\mathfrak{pol}$ la politique définie dans l'exemple précédent. $[\![\mathfrak{pol}]\!]$ contient les atomes $\mathbf{read}(\mathbf{Alice}, \mathbf{file}(\mathbf{succ}^{2k}(\mathbf{zero})))$, $\mathbf{write}(\mathbf{Bob}, \mathbf{file}(\mathbf{succ}^{3k}(\mathbf{zero})))$, $\mathbf{read}(\mathbf{Charlie}, \mathbf{file}(\mathbf{succ}^{2k}(\mathbf{zero})))$ et $\mathbf{write}(\mathbf{Charlie}, \mathbf{file}(\mathbf{succ}^{3k}(\mathbf{zero})))$, pour tout $k \in \mathbb{N}$.

Proposition 3.19. Pour toute politique de sécurité $\mathfrak{pol}$, l'ensemble $[\![\mathfrak{pol}]\!]$ est effectivement fortement régulier.

❝ *Démonstration.* Considérons un ensemble (éventuellement ordonné) de règles de sécurité $\Gamma_{\mathfrak{mod}} = (action_i \mapsto \varphi_i)_{i=1,\ldots,n}$. Notons, pour tout $I = \{i_1, \ldots, i_q\} \subseteq [1,n]$ et $i \in [1,n]$:

- $rec_I = \bigcap_{i\in I} rec(action_i) \cap \bigcup_{i\in[1,n]\setminus I} \overline{rec(action_i)}$
- $rec_i^{fst} = rec(action_i) \cap \bigcap_{j<i} \overline{rec(action_j)}$
- ψ_i la formule $\exists y_1, \ldots, y_k : x_1 = t_1 \wedge \ldots \wedge x_m = t_m \wedge \varphi_i[y_1, \ldots, y_k]$ où $action_i = \mathbf{act}(t_1, \ldots, t_m), \mathbf{act} \in \mathcal{P}$, $\{y_1, \ldots, y_k\} = \mathcal{V}ar(action_i)$. $x_1, \ldots, x_m$ sont les nouvelles variables du motif d'action.

On redéfinit $\Gamma^{\zeta}_{\mathfrak{mod}}$ de la façon suivante (cette nouvelle définition « regroupe » les éléments ayant le même comportement relativement à la fonction $\Gamma^{\zeta}_{\mathfrak{mod}}$ pour pouvoir en donner une caractérisation finie) :

- si $\zeta = \texttt{and}$, alors $\Gamma^{\zeta}_{\mathfrak{mod}}$ est la fonction qui à tout $rec_I \neq \varnothing$ associe $\bigwedge_{i\in I} \psi_i$, et associe $\top$ à $\mathcal{A}ct(\tilde{\Sigma}) \setminus \bigcup_{i=1}^{n} rec(action_i)$,
- si $\zeta = \texttt{or}$, alors $\Gamma^{\zeta}_{\mathfrak{mod}}$ est la fonction qui à tout $rec_I \neq \varnothing$ associe $\bigvee_{i\in I} \psi_i$, et associe $\bot$ à $\mathcal{A}ct(\tilde{\Sigma}) \setminus \bigcup_{i=1}^{n} rec(action_i)$,
- si ζ is $\texttt{or_else}^{\nu}$, alors $\Gamma^{\zeta}_{\mathfrak{mod}}$ associe ψ_i à rec_i^{fst} et ν à $\mathcal{A}ct(\tilde{\Sigma}) \setminus \bigcup_{i=1}^{n} rec(action_i)$.

$\Gamma^{\zeta}_{\mathfrak{mod}}$ doit se comprendre de la façon suivante : étant donnée une action (atome clos) $ac = \mathbf{act}(t_1, \ldots, t_m)$ appartenant à $\mathcal{L} \in \mathcal{D}om(\Gamma^{\zeta}_{\mathfrak{mod}})$ (un tel $\mathcal{L}$ existe et est unique puisque le domaine de $\Gamma^{\zeta}_{\mathfrak{mod}}$ forme une partition de l'ensemble des actions), l'action ac est permise ssi $(t_1, \ldots, t_m)$ est une solution de $\Gamma^{\zeta}_{\mathfrak{mod}}(\mathcal{L})$. Formellement, l'automate reconnaissant $[\![\mathfrak{pol}]\!]$ est le résultat de l'opération suivante :

$$\bigcup_{\mathcal{L}\in\mathcal{D}om(\Gamma^{\zeta}_{\mathfrak{mod}})} \left(\mathcal{L} \cap \mathcal{S}ol_{\rho}(\Gamma^{\zeta}_{\mathfrak{mod}}(\mathcal{L})) \right)$$

❞

En conséquence de la proposition précédente, on obtient la faculté d'« interroger » la politique. Interroger une politique consiste à calculer le résultat de requêtes de la forme « Y a-t-il au moins un utilisateur pouvant accéder au fichier f ? » ou « Tous les utilisateurs peuvent-ils accéder à au moins un fichier ? ». Dans la littérature, cette technique est parfois appelée « what-if analysis » ou « administrator queries ». Dans [Kirchner et al., 2009], il est montré comment résoudre des requêtes exprimées sous la forme d'une action avec variables lorsque la politique est exprimée au moyen d'un système de réécriture. Dans ce chapitre, nous considérons une classe plus large de requêtes.

Définition 3.20. Étant donnée une politique $\mathfrak{pol}$ sur $\tilde{\Sigma}$, on appelle **requête** sur $\mathfrak{pol}$ toute formule du premier ordre sur $\tilde{\Sigma}$.

Exemple 3.21. Soit $\mathfrak{pol}$ la politique définie dans les exemples précédents. La formule suivante est une requête sur $\mathfrak{pol}$ caractérisant les utilisateurs qui ne peuvent effectuer aucune action.

$$Q(s) \triangleq \forall o, \neg\mathbf{read}(s, o) \wedge \neg\mathbf{write}(s, o)$$

Les solutions de la requête dans $\mathfrak{pol}$, notée $\mathcal{S}ol_\rho(Q)$, est vide puisque tous les utilisateurs (Alice, Bob et Charlie) peuvent lire ou écrire au moins un fichier.

Proposition 3.22. Étant donnée une politique $\mathfrak{pol}$, l'ensemble $\mathcal{S}ol_\rho(Q)$ est effectivement fortement régulier pour toute requête Q.

Il s'agit d'une conséquence directe de la proposition 3.19 et du théorème 3.16. Cette proposition, très puissante en ce sens où elle nous affirme que l'on peut évaluer toute requête du premier ordre sur la sémantique d'une politique donnée, justifie le choix du fragment $\mathcal{H}_1$ pour la spécification de configurations de sécurité. De plus, le pouvoir d'expression offert par ce fragment est suffisament important pour pouvoir décrire les cas réels de configuration.

3.3 Changement de modèle

3.3.1 Positionnement du problème

Pour permettre la maintenabilité d'une politique, pour simplifier son administration, pour rendre moins complexe sa compréhension ou pour augmenter le degré de granularité dans le calcul des autorisations, on peut être amené à devoir ré-exprimer une politique dans un nouveau modèle de sécurité. Cette étape de traduction est difficile et dangereuse. Nous proposons ici une technique de transformation automatique que l'on pourrait également appeler d'« auto-configuration ». La méthode proposée est basée sur un algorithme de construction de modèles logiques réguliers [Peltier, 2009].

Exprimons formellement le problème que nous nous proposons d'étudier. Étant donnés une politique $\mathfrak{pol}_1 = \langle\, \mathfrak{mod}_1^{\zeta_1}, \rho_1^{\kappa_1} \,\rangle$ sur une signature $\tilde{\Sigma}$ ainsi qu'un modèle de

sécurité $\mathfrak{mod}_2^{\zeta_2}$ et une extension de signature κ_2, on cherche à calculer une configuration ρ_2 telle que $\langle\, \mathfrak{mod}_1^{\zeta_1}, \rho_1^{\kappa_1} \,\rangle \approx \langle\, \mathfrak{mod}_2^{\zeta_2}, \rho_2^{\kappa_2} \,\rangle$. On notera ce problème sous la forme d'une équation : $\langle\, \mathfrak{mod}_2^{\zeta_2}, X^{\kappa_2} \,\rangle \approx \langle\, \mathfrak{mod}_1^{\zeta_1}, \rho_1^{\kappa_1} \,\rangle$ où X est la variable de l'équation.

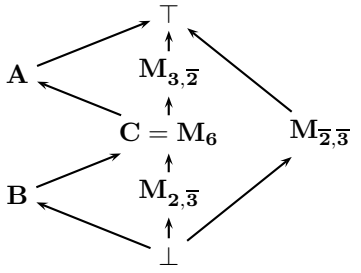
Exemple 3.23. Considérons un nouveau modèle de sécurité, basé sur la notion de treillis de sécurité, que nous noterons $\mathfrak{lbac}$. L'idée consiste à assigner chaque sujet et chaque fichier à un niveau de sécurité et à ordonner ces niveaux en treillis. Soit donc $\Sigma_{\mathfrak{lbac}}$ étendant Σ avec la sorte Level, les symboles définis $\mathbf{f_S} : \mathsf{Subject} \to \mathsf{Level}$ (représentant l'assignation des niveaux aux sujets), $\mathbf{f_O} : \mathsf{Object} \to \mathsf{Level}$, (représentant l'assignation des niveaux aux fichiers) et le prédicat $\mathbf{inf} : \mathsf{Level} \times \mathsf{Level}$ représentant le treillis sur ces niveaux. Puisque nous souhaitons que le symbole **inf** soit interprété par un treillis, nous munissons le modèle de la théorie $\mathscr{T}_{\mathfrak{lbac}}$ suivante :

$$\begin{cases} \forall x : \mathbf{inf}(x, x) \\ \forall x, y : \mathbf{inf}(x, y) \wedge \mathbf{inf}(y, z) \Rightarrow \mathbf{inf}(x, z) \\ \forall x, y : \mathbf{inf}(x, y) \wedge x \neq y \Rightarrow \neg\mathbf{inf}(y, x) \\ \exists x_{max}, x_{min}, \forall x, \mathbf{inf}(x, x_{max}) \wedge \mathbf{inf}(x_{min}, x) \end{cases}$$

Le modèle $\mathfrak{lbac}$ que l'on considère applique les principes que nous avons déjà évoqués de "no read up" et "no write down", ce qui consiste à définir $\Gamma_{\mathfrak{lbac}}$ par les règles suivantes :

$$\begin{cases} \mathbf{read}(s, o) & \mapsto & \mathbf{inf}(\mathbf{f_O}(o), \mathbf{f_S}(s)) \\ \mathbf{write}(s, o) & \mapsto & \mathbf{inf}(\mathbf{f_S}(s), \mathbf{f_O}(o)) \end{cases}$$

Enfin, la définition des niveaux est déférée au moment de la configuration et donc $\mathcal{S}^+_{\mathfrak{lbac}} = \{\mathsf{Level}\}$. « Transformer la politique $\mathfrak{pol} = \langle\, \mathfrak{rbac}, \rho^\kappa \,\rangle$ (de l'exemple 3.13) vers le modèle $\mathfrak{lbac}$ » consiste à déterminer une configuration interprétant les symboles $\mathbf{inf}, \mathbf{f_O}$ et $\mathbf{f_S}$ engendrant avec le modèle $\mathfrak{lbac}$ exactement le même ensemble d'autorisations que $\langle\, \mathfrak{rbac}, \rho^\kappa \,\rangle$. Il s'agit donc de déterminer l'affectation des niveaux aux utilisateurs et aux fichiers ($\mathbf{f_O}$ et $\mathbf{f_S}$) et d'ordonner ces fichiers (**inf**). Une solution, si κ contient huit constantes de sorte Level, est décrite par la figure suivante :



$$\begin{cases} \mathbf{f_S}(\mathbf{Alice}) = \mathbf{A} & \Leftarrow & \\ \mathbf{f_S}(\mathbf{Bob}) = \mathbf{B} & \Leftarrow & \\ \mathbf{f_S}(\mathbf{Charlie}) = \mathbf{C} & \Leftarrow & \\ \mathbf{f_O}(\mathbf{file}(n)) = \mathbf{C} & \Leftarrow & \mathbf{rem_0^6}(n) \\ \mathbf{f_O}(\mathbf{file}(n)) = \mathbf{M_{3,\overline{2}}} & \Leftarrow & \mathbf{rem_0^3}(n), \mathbf{rem_1^2}(n) \\ & \vdots\Leftarrow & \\ \mathbf{inf}(\bot, \mathbf{B}) & \Leftarrow & \end{cases}$$

où A (resp. B, resp. C) représente le niveau d'Alice (resp. Bob, resp. Charlie), les flèches décrivent l'ordre interprétant **inf** et M_n est le niveau affecté aux fichiers identifiés par un nombre multiple (resp. non multiple) de n si n n'est pas (resp. est) surligné.

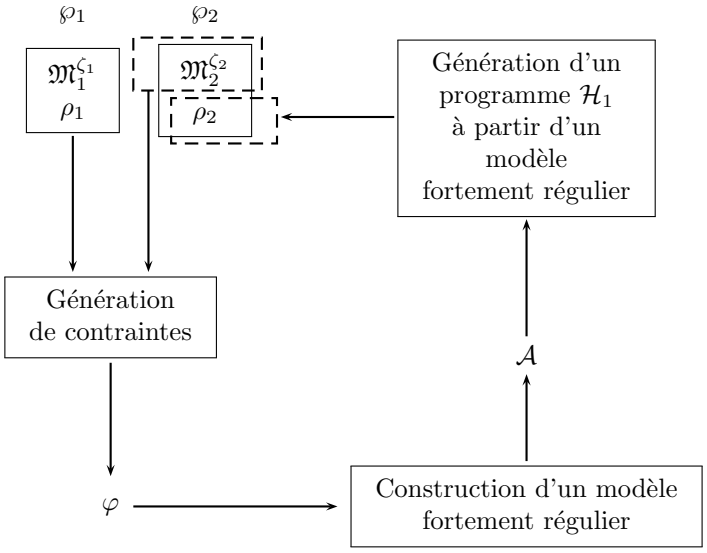


FIGURE 3.1 – Schéma de l'algorithme de changement de modèle

3.3.2 Méthode

Pour construire la politique cible, nous générons un ensemble de contraintes exprimant le fait que sa sémantique doit être celle de la politique d'origine. Par ailleurs, sachant que les configurations décrivent des interprétations logiques fortement régulières et que l'algorithme proposé dans [Peltier, 2009] engendre des interprétations régulières, des contraintes supplémentaires seront ajoutées de telle sorte que toute interprétation régulière validant les contraintes soit fortement régulière. Enfin, une dernière étape consistera à calculer un programme $\mathcal{H}_1$ à partir d'une interprétation fortement régulière. L'algorithme général est illustré par la figure 3.1.

Génération des contraintes. Nous allons décrire les contraintes dont les modèles logiques correspondent aux interprétations des symboles du modèle cible engendrant exactement la même sémantique que la politique d'origine. Puisque cette dernière est fortement régulière, il faut et il suffit de savoir axiomatiser l'ensemble des éléments appartenant à un ensemble régulier. Plus précisément, étant donné un automate $\mathbb{A} = \langle \Sigma, Q, F, \Delta \rangle$, on note $Ax(\mathbb{A})[x]$ la formule telle que pour tout terme clos t, $Ax(\mathbb{A})[t]$ est un théorème du premier ordre (c.-à-d. valide dans toute interprétation) ssi t est reconnu par $\mathbb{A}$ et est contradictoire (c.-à-d. valide dans aucune interprétation) sinon. Étant entendu que tous les automates considérés sont complets, déterministes et minimaux, $Ax(\mathbb{A})[x] = Ax_\Delta(\mathbb{A}) \wedge Ax_F(\mathbb{A})[x]$, où $Ax_\Delta(\mathbb{A})$ est donnée par la formule suivante :

$$\bigwedge_{\mathbf{f}(q_1,\ldots,q_n)\to_\Delta q} \left(\forall y_1, \ldots, y_n : \bigwedge_{i=1}^{n} q_i(y_i) \Rightarrow q(\mathbf{f}(y_1, \ldots, y_n)) \right) \wedge \bigwedge_{q \neq q' \in Q^2} \left(\forall x : q(x) \Rightarrow \neg q'(x) \right)$$

et $Ax_F(\mathbb{A})[x] = \bigvee_{q_F \in F} q_F(x)$.

Intuitivement, à chaque état q de l'automate on associe un prédicat $\mathbf{q}$ dont l'interprétation doit être l'ensemble des termes clos se réduisant (pour la relation de transition de l'automate) à q. Autrement dit, $\mathbf{q}(t)$ est vrai ssi $t \xrightarrow{*}_{\Delta} q$.

> *Exemple 3.24.* Si l'on considère l'automate reconnaissant l'ensemble des entiers pairs $\mathbb{A}_{even} = (\{q_{odd}, q_{even}\}, \{\mathbf{zero} :\to \mathsf{Nat}, \mathbf{succ} : \mathsf{Nat} \to \mathsf{Nat}\}, \{q_{even}\}, \Delta)$ avec :
>
> $$\Delta = \begin{cases} \mathbf{zero} & \to & q_{even} \\ \mathbf{succ}(q_{even}) & \to & q_{odd} \\ \mathbf{succ}(q_{odd}) & \to & q_{even} \end{cases}$$
>
> alors $Ax_\Delta(\mathbb{A}_{even})$ est donnée par :
>
> $$\begin{array}{ll} & \mathbf{q_{even}}(\mathbf{zero}) \\ \wedge & \forall x : \mathbf{q_{even}}(x) \Rightarrow \mathbf{q_{odd}}(\mathbf{succ}(x)) \\ \wedge & \forall x : \mathbf{q_{odd}}(x) \Rightarrow \mathbf{q_{even}}(\mathbf{succ}(x)) \\ \wedge & \forall x : \mathbf{q_{even}}(x) \Rightarrow \neg\mathbf{q_{odd}}(x) \\ \wedge & \forall x : \mathbf{q_{odd}}(x) \Rightarrow \neg\mathbf{q_{even}}(x) \end{array}$$
>
> et $Ax_F(\mathbb{A}_{even})[x] = \mathbf{q_{even}}(x)$. La première (resp. deuxième, resp. troisième) sous-formule de $Ax_\Delta(\mathbb{A}_{even})$ est produite à partir de la première (resp. deuxième, resp. troisième) règle de l'automate $\mathbb{A}_{even}$.

Considérons désormais une politique de sécurité cohérente $\langle\, \mathfrak{mod}_1^{\zeta_1}, \rho_1^{\kappa_1} \,\rangle$ sur $\tilde{\Sigma}$ ainsi qu'un modèle de sécurité $\mathfrak{mod}_2^{\zeta_2}$, sur cette même signature, munie d'une extension κ_2 de $\Sigma_{\mathfrak{mod}_2}$. Rappelons que pour tout modèle $\mathfrak{mod}^{\zeta}$, le domaine de $\Gamma^{\zeta}_{\mathfrak{mod}}$ forme une partition $\{\mathcal{L}_1, \ldots, \mathcal{L}_m\}$ de l'ensemble des actions $\mathcal{A}t(\tilde{\Sigma})$ (*c.f.* démonstration de la proposition 3.19). Autrement dit, pour toute action $act = \mathbf{act}(t_1, \ldots, t_n)$, il existe exactement un i tel que $act \in \mathcal{L}_i$. La contrainte permettant d'évaluer l'autorisation de l'action act est alors donnée par $\varphi[x_1, \ldots, x_n] = \Gamma^{\zeta}_{\mathfrak{mod}}(\mathcal{L}_i)$. Plus précisément, l'action act est permise relativement à la configuration ρ si et seulement si $\rho \models \varphi[t_1, \ldots, t_n]$. A fortiori, l'ensemble des langages $\mathcal{L}_1 \cap \mathcal{L}_2$ tels que $(\mathcal{L}_1 \times \mathcal{L}_2) \in \mathcal{D}om(\Gamma^{\zeta_1}_{\mathfrak{mod}_1}) \times \mathcal{D}om(\Gamma^{\zeta_2}_{\mathfrak{mod}_2})$ et $\mathcal{L}_1 \cap \mathcal{L}_2 \neq \varnothing$ forment également une partition de $\mathcal{A}t(\tilde{\Sigma})$. À noter que tous les éléments de $\mathcal{L}_1 \cap \mathcal{L}_2$ possèdent nécessairement le même symbole d'action de tête. Dans ce cadre, la formule $Ax(\mathcal{L}_1 \cap \mathcal{L}_2)[\mathbf{act}(x_1, \ldots, x_n)]$ caractérise l'ensemble des valeurs de $x_1, \ldots, x_n$ pour lesquelles $\mathbf{act}(x_1, \ldots, x_n)$ appartient à $\mathcal{L}_1 \cap \mathcal{L}_2$. $Ax\left(\mathcal{S}ol_{\rho_1}\left(\Gamma^{\zeta_1}_{\mathfrak{mod}_1}(\mathcal{L}_1)\right)\right)[\#(x_1, \ldots, x_n)]$ caractérise quant à elle l'ensemble des valeurs de $x_1, \ldots, x_n$ pour lesquelles $\#(x_1, \ldots, x_n)$ est une solution dans ρ_1 des contraintes associées à $\mathcal{L}_1$ donc telles que $\mathbf{act}(x_1, \ldots, x_n)$ est une action permise par la politique $\langle\, \mathfrak{mod}_1^{\zeta_1}, \rho_1^{\kappa_1} \,\rangle$. Il est nécessaire de bien comprendre que quels que soient les termes clos $t_1, \ldots, t_n$, la validité de la formule $Ax\left(\mathcal{S}ol_{\rho_1}\left(\Gamma^{\zeta_1}_{\mathfrak{mod}_1}(\mathcal{L}_1)\right)\right)[\#(t_1, \ldots, t_n)]$ ne dépend pas de l'interprétation dans laquelle elle est évaluée. Autrement dit, $\varnothing \models Ax\left(\mathcal{S}ol_{\rho_1}\left(\Gamma^{\zeta_1}_{\mathfrak{mod}_1}(\mathcal{L}_1)\right)\right)[\#(t_1, \ldots, t_n)]$ si l'action $\mathbf{act}(t_1, \ldots, t_n)$ est autorisée par la politique $\langle\, \mathfrak{mod}_1^{\zeta_1}, \rho_1^{\kappa_1} \,\rangle$ et $\varnothing \models \neg Ax\left(\mathcal{S}ol_{\rho_1}\left(\Gamma^{\zeta_1}_{\mathfrak{mod}_1}(\mathcal{L}_1)\right)\right)[\#(t_1, \ldots, t_n)]$

sinon. Enfin, $\Gamma^{\zeta_2}_{\mathfrak{mod}_2}(\mathcal{L}_2)[x_1,\ldots,x_n]$ est la contrainte à satisfaire pour que l'action $\mathbf{act}(x_1,\ldots,x_n)$ soit permise selon le modèle $\mathfrak{mod}_2{}^{\zeta_2}$ relativement à l'interprétation des prédicats de $\tilde{\Sigma}_{\mathfrak{mod}_2}$.

En notant **act** le symbole de tête des actions de $\mathcal{L}_1$ et $\mathcal{L}_2$, on note $Cons(\mathcal{L}_1,\mathcal{L}_2)$ la formule suivante :

$$\begin{aligned} & Ax_\Delta\left(\mathcal{S}ol_{\rho_1}\left(\Gamma^{\zeta_1}_{\mathfrak{mod}_1}(\mathcal{L}_1)\right)\right) \wedge Ax_\Delta(\mathcal{L}_1\cap\mathcal{L}_2) \\ \wedge \quad & \Big\{\forall x_1,\ldots,x_n : Ax_F(\mathcal{L}_1\cap\mathcal{L}_2)[\mathbf{act}(x_1,\ldots,x_n)] \\ & \Rightarrow \left(Ax_F\left(\mathcal{S}ol_{\rho_1}\left(\Gamma^{\zeta_1}_{\mathfrak{mod}_1}(\mathcal{L}_1)\right)\right)[\#(x_1,\ldots,x_n)] \Leftrightarrow \Gamma^{\zeta_2}_{\mathfrak{mod}_2}(\mathcal{L}_2)[x_1,\ldots,x_n]\right)\Big\} \end{aligned}$$

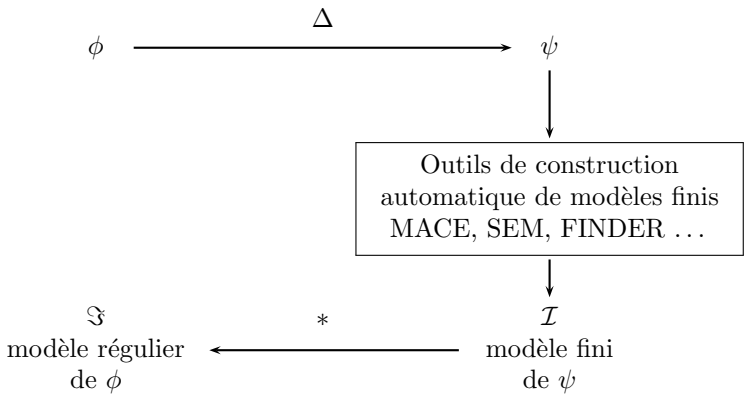
$Cons(\mathcal{L}_1,\mathcal{L}_2)$ est donc une théorie caractérisant l'équivalence entre la satisfiabilité de la contrainte associée à chaque action par $\mathfrak{mod}_2$ (relativement à l'interprétation des prédicats de $\tilde{\Sigma}_{\mathfrak{mod}_2}$) avec la satisfiabilité de la contrainte associée à cette même action par $\mathfrak{mod}_1$ relativement à ρ_1.

Enfin, on note $Cons\left(\langle\, \mathfrak{mod}_2^{\zeta_2}, X^{\kappa_2}\,\rangle \approx \langle\, \mathfrak{mod}_1^{\zeta_1}, \rho_1^{\kappa_1}\,\rangle\right)$ la formule suivante :

$$\mathscr{T}_{\mathfrak{mod}_2} \wedge \bigwedge_{(\mathcal{L}_1,\mathcal{L}_2)\in \mathcal{D}om(\Gamma^{\zeta_1}_{\mathfrak{mod}_1})\times\mathcal{D}om(\Gamma^{\zeta_2}_{\mathfrak{mod}_2})} Cons(\mathcal{L}_1,\mathcal{L}_2)$$

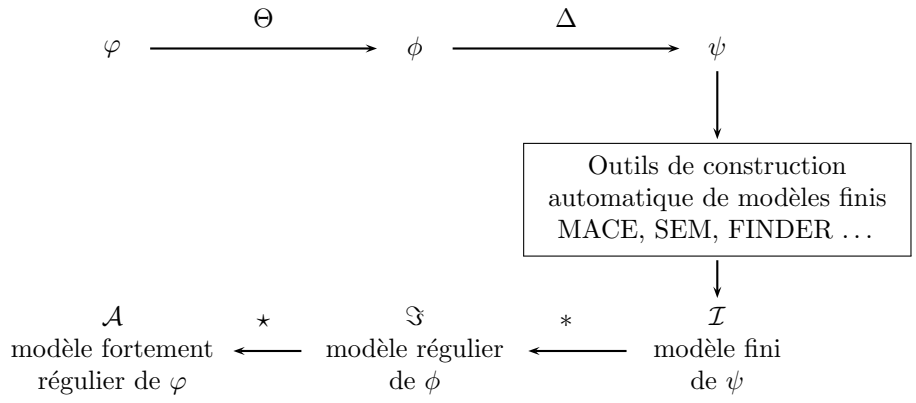
Cette formule est une théorie dont les modèles logiques interprètent les symboles de prédicats de $\tilde{\Sigma}_{\mathfrak{mod}_2}$ de telle sorte que les autorisations calculées à partir de $\mathfrak{mod}_2$ et de ces modèles logiques correspondent à l'ensemble des autorisations établies par la politique d'origine. À chaque modèle logique fortement régulier (c.-à-d. interprétant les prédicats par des ensembles et relations fortement réguliers) correspond exactement une configuration. Cependant, la recherche d'interprétations régulières de cette théorie peut engendrer des modèles logiques non fortement réguliers, donc ne correspondant pas à une configuration. C'est pourquoi nous allons transformer cette théorie de telle sorte que toutes les interprétations régulières de la théorie soit fortement régulières.

Construction d'un modèle logique. Dans [Peltier, 2009], Peltier propose une méthode pour utiliser les outils de construction automatiques de modèles (logiques) finis dans l'objectif de construire des modèles infinis d'une formule donnée. Les modèles ainsi construits sont des interprétations de Herbrand dans lesquelles les symboles de prédicats sont interprétés par des relations régulières (spécifiées par des automates d'arbres). Son approche est fondée sur une transformation de la formule dont on recherche un modèle logique : une formule ϕ est transformée en une formule $\Delta(\phi)$ telle que ϕ possède un modèle $\Im$ représentable par des automates de tuples de termes si et seulement si $\Delta(\phi)$ possède un modèle fini $\mathcal{I}$. Il établit ensuite une application $*$ permettant d'obtenir $\Im$ à partir de $\mathcal{I}$. Il a montré la correction et la complétude de son algorithme dans le sens suivant : un modèle régulier de Herbrand $\Im$ de φ existe si et seulement si il existe un modèle fini $\mathcal{I}$ de $\Delta(\phi)$ (et alors $\Im = \mathcal{I}^*$). Le principe général de son algorithme est illustré par le schéma suivant :



Dans le contexte qui est le nôtre, il est nécessaire de construire à partir d'une formule φ sur $\tilde{\Sigma}$ une interprétation syntaxique de domaine $\mathcal{T}(\Sigma\mathscr{C}ons)$ associant à chaque prédicat une relation fortement régulière. En effet, nous avons vu que l'ensemble des configurations d'un modèle donné $\mathfrak{mod}$ sur une signature de sécurité $\tilde{\Sigma}$ est isomorphe à l'ensemble des interprétations syntaxiques fortement régulières de $\tilde{\Sigma}_{\mathfrak{mod}}$ de domaine $\Sigma\mathscr{C}ons_{\mathfrak{mod}} = \Sigma\mathscr{C}ons$ (d'après le théorème 3.11 et sa réciproque qui est triviale). Il nous faut donc étendre la méthode de [Peltier, 2009]. Nous nous proposons, suivant l'idée de Peltier, de transformer notre formule d'origine φ en une formule $\Theta(\varphi)$ telle que $\mathcal{A}$ est une interprétation fortement régulière de $\tilde{\Sigma}$ de domaine $\mathcal{T}(\Sigma\mathscr{C}ons)$ modèle de φ si et seulement si il existe un modèle de Herbrand régulier $\Im$ de $\Theta(\varphi)$ (et alors $\mathcal{A} = \Im^\star$ pour une certaine application $\star$).

Le principe de la méthode est alors illustré par le schéma suivant :



Tout d'abord, remarquons que les relations unaires régulières sont fortement régulières. Puisque pour toute relation n-aire fortement régulière R, il existe un ensemble (relation unaire) régulier $R^{\#}$ contenant les termes $\{\#(t_1, \ldots, t_n) \mid (t_1, \ldots, t_n) \in R\}$ et réciproquement, alors il faut et il suffit de transformer tous les prédicats n-aires en prédicats unaire pour que l'ensemble des interprétations régulières de ce prédicat soit exactement l'ensemble des interprétations fortement régulières du prédicat.

Par ailleurs, comme la méthode de [Peltier, 2009] ne considère que des interprétations de Herbrand, il est nécessaire de considérer les symboles définis comme des symboles de prédicats particuliers. Ce processus est standard et est souvent appelé « flattening ». Chaque symbole défini $\mathbf{f}$ est donc remplacé par un nouveau symbole de prédicat $\mathbf{f}'$ tel que $\mathbf{f}'(\#(t_1,\ldots,t_n,u))$ est vrai si et seulement si $\mathbf{f}(t_1,\ldots,t_n)=u$ l'est. Pour que $\mathbf{f}'$ soit toujours interprété par une fonction, il est nécessaire d'ajouter un axiome établissant que pour tout n-uplet de valeurs $x_1,\ldots,x_n$, il existe exactement une valeur y telle que $\mathbf{f}'(\#(x_1,\ldots,x_n,y))$. Notez bien que l'utilisation du symbole $\#$ est un abus de notation et qu'il faut en réalité considérer un symbole $\#^n$ pour tout n ($\#^n$ étant le symbole de tête des n-uplets). Par ailleurs, les signatures considérées dans [Peltier, 2009] sont mono-sortées. En conséquence, il est nécessaire d'ajouter une axiomatisation des profils des prédicats et symboles de fonction définis. En résumé, étant donnée une formule φ sur $\tilde{\Sigma}$, $\Theta(\varphi)$ est obtenue en appliquant les transformations suivantes :

- chaque atome $\mathbf{p}(t_1,\ldots,t_n)$ est remplacé par l'atome $\mathbf{p}(\#(t_1,\ldots,t_n))$,
- tout terme t comportant au moins un symbole défini et apparaissant dans un atome (ou une égalité) est remplacé par une variable x existentiellement quantifiée et l'égalité $t=x$ est ajoutée,
- toute égalité de la forme $\mathbf{f}(t_1,\ldots,t_n)=x$ est remplacée par $\mathbf{f}'(\#(t_1,\ldots,t_n,x))$,
- pour toute sorte s, on ajoute (sous la forme d'une conjonction) $Ax_\Delta(\mathcal{T}(\Sigma\mathscr{C}ons)_\mathsf{s})$ (où $\mathcal{T}(\Sigma\mathscr{C}ons)_\mathsf{s}$ vu comme l'automate reconnaissant les termes de donnée de sorte s),
- pour tout prédicat $\mathbf{p}:\mathsf{s}_1\times\ldots\times\mathsf{s}_n$, on ajoute (sous la forme d'une conjonction) la formule :

$$\forall x_1,\ldots,x_n,\mathbf{p}(\#(x_1,\ldots,x_n))\Rightarrow\bigwedge_{i=1}^{n}Ax_F(\mathcal{T}(\Sigma\mathscr{C}ons)_{\mathsf{s}_i})[x_i]$$

  (on fait de même pour les symboles définis vus comme des prédicats),
- pour tout symbole defini $\mathbf{f}:\mathsf{s}_1\times\ldots\times\mathsf{s}_n\to\mathsf{s}$, on ajoute (sous la forme d'une conjonction) les formules :

$$\forall x_1,\ldots,x_n,y_1,y_2,\ \mathbf{f}'(\#(x_1,\ldots,x_n,y_1))\wedge\mathbf{f}'(\#(x_1,\ldots,x_n,y_2))\Rightarrow y_1=y_2$$

  indiquant que $\mathbf{f}'$ doit être interprété par une relation fonctionnelle et

$$\begin{array}{l}\forall x_1,\ldots,x_n,\bigwedge_{i=1}^{n}Ax_F(\mathcal{T}(\Sigma\mathscr{C}ons)_{\mathsf{s}_i})[x_i]\\ \quad\Rightarrow\big(\exists y,Ax_F(\mathcal{T}(\Sigma\mathscr{C}ons)_\mathsf{s})[y]\wedge\mathbf{f}'(\#(x_1,\ldots,x_n,y))\big)\end{array}$$

  indiquant que $\mathbf{f}'$ doit être interprétée par une application (fonction totale).

La formule obtenue est donc exprimée sur une signature dont les prédicats sont les prédicats de $\tilde{\Sigma}$ et les symboles fonctionnels de $\Sigma\mathscr{D}ef$ ainsi que quelques prédicats auxiliaires (correspondant notamment aux états de tous les automates d'arbres impliqués) et dont les termes sont construits sur $\Sigma\mathscr{C}ons$. Une interprétation de Herbrand de cette signature induit donc une unique interprétation syntaxique de $\tilde{\Sigma}$ basée sur les constructeurs. Pour obtenir l'inteprétation fortement régulière d'une

formule φ à partir de l'interprétation régulière modèle de $\Theta(\varphi)$, il suffit de modifier chaque automate associé à un symbole défini $\mathbf{f}$ reconnaissant des éléments de la forme $\#(t_1, \ldots, t_n, t)$ par un automate reconnaissant les termes correspondant sous la forme $\#(\mathbf{f}(t_1, \ldots, t_n), t)$. L'union de ces automates décrira l'interprétation du prédicat d'égalité. Les automates obtenus pour décrire les interprétations des prédicats de $\tilde{\Sigma}$ restent inchangés.

Si l'on retourne à notre problème initial, notre objectif est de calculer, étant donnés deux modèles $\mathfrak{mod}_1^{\zeta_1}$ et $\mathfrak{mod}_2^{\zeta_2}$ sur une même signature de sécurité $\tilde{\Sigma}$ ainsi qu'une configuration ρ_1, une interprétation syntaxique fortement régulière (basée sur les constructeurs) de $\tilde{\Sigma}_{\mathfrak{mod}_2}$ modèle de $Cons\left(\langle\, \mathfrak{mod}_2^{\zeta_2}, X^{\kappa_2} \,\rangle \approx \langle\, \mathfrak{mod}_1^{\zeta_1}, \rho_1^{\kappa_1} \,\rangle\right)$. L'interprétation obtenue correspond à une configuration ρ_2 solution de l'équation $\langle\, \mathfrak{mod}_2^{\zeta_2}, X^{\kappa_2} \,\rangle \approx \langle\, \mathfrak{mod}_1^{\zeta_1}, \rho_1^{\kappa_1} \,\rangle$. Pour être précis, la solution est un programme $\mathcal{H}_1$ dont la sémantique est l'intepétation obtenue (la construction d'un programme $\mathcal{H}_1$ à partir des automates décrivant les interprétations des prédicats est triviale et est décrite dans [Nieslon et al., 2002]).

3.4 Travaux reliés

De nombreux cadres ont été définis pour permettre l'analyse des politiques de sécurité. Le lecteur pourra consulter [Damianou et al., 2002] pour obtenir un premier aperçu des méthodes proposées dans la littérature. Notre approche s'inscrit dans la lignée de l'utilisation des méthodes algébriques déclaratives, regroupant notamment les formalismes fondés sur des fragments de la logique du premier ordre et ceux fondés sur les systèmes de réécriture.

Approches fondées sur la logique. Les logiques utilisées pour la spécification de politiques de sécurité sont souvent non-monotones, c.-à-d. sont telles que l'ensemble des conséquences logiques d'une théorie ne croît pas avec le nombre d'axiomes de cette dernière. L'utilisation de telles logiques est motivée par la nécessité de prendre une décision en l'absence d'information. La première utilisation d'une logique non monotone pour la spécification de politiques a été proposée dans [Woo and Lam, 1993]. Dans cet article, les auteurs montrent que la logique des défauts permet d'exprimer facilement un certain nombre de politiques de sécurité. Ils illustrent leur propos en présentant une axiomatisation du modèle de Bell et Lapadula dans leur formalisme. Les autorisations sont exprimées par les atomes de la forme $\mathbf{auth}(user, action, object)$ et, en notant $\top$ l'atome toujours vrai, les règles de sécurité exprimant le comportement par défaut d'une politique dite ouverte (resp. fermée) sont spécifiées de la façon suivante :

$$\frac{\top : \mathbf{auth}(x, y, z)}{\mathbf{auth}(x, y, z)} \qquad \frac{\top : \neg\mathbf{auth}(x, y, z)}{\neg\mathbf{auth}(x, y, z)}$$

où $\dfrac{P : Q_1, \ldots, Q_n}{Q}$ signifie « si P est dérivable et les Q_i satisfiables, alors on dérive Q ». Dans notre approche, le calcul de l'autorisation par défaut est donné au moyen d'une stratégie d'interprétation.

Dans [Halpern and Weissman, 2008], les auteurs proposent de spécifier un modèle de sécurité au moyen d'une (unique) règle se présentant sous l'une des deux formes suivantes :

$$\begin{array}{rlll} & \forall x_1, \ldots, x_n, & (\varphi & \Rightarrow \quad \mathbf{permitted}(t_{subject}, t_{action})) \\ \text{ou} & \forall x_1, \ldots, x_n, & (\varphi & \Rightarrow \quad \neg\mathbf{permitted}(t_{subject}, t_{action})) \end{array}$$

où φ est une formule du premier ordre quelconque, $t_{subject}$ un terme de sorte Subject et t_{action} un terme de sorte Action. La notion de configuration apparaît également (sous la terminologie d'environnement [2]). Les auteurs proposent de décrire une configuration sous la forme d'une formule du premier ordre fermée (au sens où toutes ses variables sont liées) ne contenant pas le prédicat **permitted**. L'évaluation d'une requête d'action dans le modèle M relativement à la configuration C est alors donnée par la résolution du problème :

$$\begin{array}{rl} & M \wedge C \models \mathbf{permitted}(t_{subject}, t_{action}) \\ \text{ou} & M \wedge C \models \neg\mathbf{permitted}(t_{subject}, t_{action}) \end{array}$$

Dans cet article, l'accent est mis sur l'expressivité, l'évaluation et la vérification de la cohérence. En revanche, les autres propriétés ainsi que l'analyse par requêtage ne sont pas mentionnées.

Un autre cadre de spécification est proposé dans [Bertino et al., 2003]. Ce dernier impose un certain nombre de notions (sujets, objets, privilèges, sessions) rendant plus spécifique l'approche proposée. Les règles d'autorisations, spécifiant si un sujet possède un privilège donné sur un objet, sont exprimées au moyen de programmes C-Datalog [Greco et al., 1992] (une extension orientée objet de Datalog [Abiteboul et al., 1995]). Des règles de contraintes, spécifiant des invariants portant sur les entités du cadre (sujets, objets, privilèges et sessions), sont également décrites par des programmes C-Datalog. Dans ces travaux, la distinction est également faite entre modèle et politique de sécurité (vue comme une instance d'un modèle). Le cadre présenté met l'accent sur les problématiques de comparaison de modèles et de vérification de la cohérence. La conversion d'une politique d'un modèle vers un autre n'est pas abordée.

Approches fondées sur la réécriture. Depuis 2006, un certain nombre de travaux a été effectué dans l'objectif de montrer l'adéquation des formalismes basés sur la réécriture avec les problématiques liées aux politiques de sécurité. Les premiers travaux publiés en ce sens ont été présentés dans [Barker and Fernández, 2006]. Dans cet article, les auteurs cherchent à démontrer que les systèmes de réécriture constituent un formalisme adaptés à l'écriture des politiques et des modèles de sécurité. Ils illustrent leur propos en proposant une spécification des modèles RBAC et ACL (Access Control List) sous la forme de règles de réécriture. Les requêtes d'accès sont représentées par des termes du premier ordre et leur évaluation par le processus de normalisation induit par le système de réécriture décrivant la politique.

Dans [Bertolissi et al., 2007], un modèle baptisé DEBAC (Dynamic Event-Based Access Control) est proposé et spécifié sous la forme d'un système de réécriture sur

2. Terme que l'on reprendra dans le chapitre suivant pour souligner son caractère dynamique.

des termes du premier ordre ainsi que des termes d'ordre supérieur du λ-calcul. La même approche est suivie dans [Barker and Fernàndez, 2007] pour la spécification du modèle ASAC (Action-Status Access Control). Le modèle est spécifié à l'aide de règles de réécriture et les propriétés fondamentales du modèle (terminaison, cohérence, complétude) sont démontrées à l'aide des techniques de réécriture. Dans la continuité de ces travaux, [Bertolissi and Fernández, 2008] formalise une théorie dans laquelle les politiques de contrôle d'accès sont combinées au moyen d'opérateurs d'ordre supérieur. Cette approche permet de bénéficier des propriétés de modularité de la théorie pour dériver la correction de la politique globale. [Bertolissi and Fernandez, 2009a] étend ces travaux en tenant compte des aspects distribués et dynamiques des environnements (distributed DBAC). Ces travaux s'inscrivent dans une approche relativement différente de la notre puisqu'ils sont dédiés à la spécification et l'analyse d'un modèle particulier.

D'autres initiatives basées sur la réécriture ont parallèlement été développées dans [Santana de Oliveira, 2007, Dougherty et al., 2007]. Ces travaux s'attachent, comme dans [Barker and Fernández, 2006], à montrer la façon dont les techniques liées à la réécriture de terme (vérification de la terminaison, de la confluence, et de la complétude suffisante) sont adaptées à la vérification de propriétés liées aux politiques de sécurité. Des extensions de la réécriture « basique » sont considérées (stratégies, réécriture modulo) pour faciliter l'écriture et la composition des politiques. Dans [Kirchner et al., 2009], une méthode est présentée pour effectuer des analyses par requêtage lorsque la politique de sécurité est exprimée sous la forme d'un système de réécriture stratégique. La procédure de vérification est basée sur le principe de narrowing, étendu par les auteurs pour tenir compte des stratégies. La méthode est illustrée par l'analyse de politiques décrivant la configuration d'un firewall.

3.5 Synthèse

Le travail présenté dans ce chapitre s'inscrit dans une démarche de développement d'un cadre formel pour spécifier, analyser et maintenir les politiques de sécurité. Nous y avons montré en quoi les spécifications des politiques de sécurité en deux parties sont particulièrement attractives, notamment en terme de réutilisabilité des spécifications. Nous avons établi plusieurs résultats et algorithmes tirant profit des avancées récentes dans les domaines des automates d'arbres et de la logique du premier ordre. Les travaux exposés dans ce chapitre correspondent à une amélioration des résultats présentés dans [Bourdier, 2011a]. Nous nous distinguons fortement de la plupart des travaux existants dans notre démarche consistant à utiliser deux langages différents pour exprimer les modèles de sécurité et les configurations. Ceci nous a permis d'adapter le pouvoir d'expression aux réels besoins de description de chaque composant de notre étude. Cette démarche rend possible la représentation de modèles complexes tout en assurant de bons résultats de décidabilité (évaluation, analyse, ...). Pour ce faire, nous avons préconisé la description des configurations de sécurité à l'aide d'un fragment plus limité de la logique du premier ordre. Ce fragment, baptisé $\mathcal{H}_1$, est de façon générale plus restrictif que les programmes Datalog comportant des négations. Cependant, l'usage de la négation n'est, à notre avis, pas

nécessaire pour décrire les configurations. En revanche, le langage que nous avons retenu pour la description des modèles de sécurité est beaucoup plus expressif que ceux proposés dans la littérature. En somme, nous avons préconisé une démarche de spécification hétérogène permettant d'être conforme aux véritables besoins de description de chaque composant tout en permettant la combinaison de ces spécifications à des fins de vérification. De surcroît, la spécification séparée de ces composants a permis d'aborder la problématique intéressante et néanmoins peu (voire pas) traitée dans la littérature de la conversion de politiques d'un modèle vers un autre.

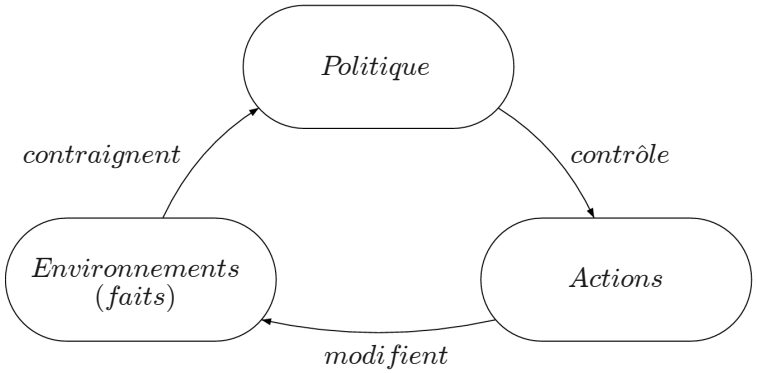
Une des limitations de l'approche proposée dans ces travaux est la supposition de l'existence d'un adminitrateur qui aurait tout pouvoir sur la configuration de la politique et qui ne serait pas lui-même soumis à une politique lorsqu'il effectue des actions administratives. Les politiques considérées sont alors « statiques » en ce sens que l'exécution d'une action autorisée par la politique n'altère pas les droits des utilisateurs. Si les tâches administratives sont vues comme des actions effectuées au sein du système d'information, alors les politiques considérées deviennent dynamiques. Une action effectuée par un utilisateur peut alors modifier l'ensemble des autorisations d'actions. Le chapitre suivant a pour objet de passer outre cette limitation.

4 Spécification et vérification de politiques dynamiques

Sommaire

4.1 Systèmes sécurisés et politiques dynamiques 72
4.1.1 Théorie de sécurité 73
4.1.2 Environnements et règles de transition 76
4.1.3 Système de sécurité 81
4.1.4 Politiques de sécurité 83
4.1.5 Système sécurisé 85
4.2 Sémantiques basées sur la réécriture 85
4.2.1 Représentation symbolique des environnements 86
4.2.2 Évaluation des contraintes 89
4.2.3 Systèmes de sécurité 93
4.2.4 Politiques de sécurité 95
4.2.5 Système sécurisé 96
4.3 Vérification 97
4.3.1 Propriétés de politiques 97
4.3.2 Le problème de l'administration 98
4.3.3 Comparaison de politique 99
4.3.4 Analyse par requêtage 100
4.4 Travaux reliés 102
4.5 Synthèse 103

Nous l'avons déjà évoqué, le cadre présenté dans le chapitre précédent est basé sur une vision statique des politiques de sécurité. Il peut être intéressant de considérer qu'une action a un impact sur la politique, ou, plus précisément, sur les informations servant de base au calcul des autorisations (la configuration de la politique). Plus généralement, on peut envisager les politiques de sécurité comme un moyen de restreindre l'évolution d'un programme. Un programme est alors constitué d'un certain nombre d'actions dont l'application modifie l'état courant du système qu'il décrit. Les politiques de sécurité délivrent les autorisations permettant au programme d'effectuer ces actions en tenant possiblement compte des informations relatives à l'état courant du système, de l'historique des actions précédemment effectuées ou encore de considérations de priorité entre les actions. Par exemple, l'autorisation d'un retrait d'argent peut dépendre du montant total disponible sur le compte correspondant mais également du montant retiré au cours des trois derniers jours. La figure suivante illustre les interactions entre les différents composants de notre étude.



Dans ce chapitre, nous proposons un cadre permettant de spécifier séparément les politiques de sécurité et les systèmes sur lesquels elles sont appliquées. Cette séparation permet non seulement d'analyser la politique indépendamment d'un quelconque système cible mais permet également d'analyser l'application d'une même politique sur des systèmes différents. Nous montrerons en quoi les systèmes de réécriture nous permettent d'obtenir de façon automatique non seulement une spécification exécutable d'un système contraint par une politique mais également des spécifications adaptées à l'analyse des politiques de sécurité et des systèmes sécurisés par une politique. Les travaux décrits dans ce chapitre ont partiellement été présentés dans [Bourdier et al., 2010b, Bourdier et al., 2010c, Bourdier et al., 2011].

4.1 Systèmes sécurisés et politiques dynamiques

Nous proposons ici un cadre dans lequel les politiques de sécurité et les systèmes sur lesquels elles sont appliquées sont spécifiées séparément. Nous nous intéressons aux systèmes dont les états évoluent en fonction de certaines actions. Puisque les notions d'état et de transition (restreinte) sont centrales, rappelons la définition classique des systèmes de transitions étiquetées.

Définition 4.1. Un **système de transitions étiquetées** S est un quadruplet (L, Q, Q_0, Δ) où :
- L est l'ensemble des étiquettes,
- Q est l'ensemble des états,
- $Q_0 \subseteq Q$ est l'ensemble des états initiaux,
- $\Delta \subseteq Q \times L \times Q$ est l'ensemble des transitions.

On note $\xrightarrow{*}_S$ la relation de transition induite par S.

Dans la présente section, nous préciserons la façon dont doivent être spécifiés les différents éléments constituant les systèmes sécurisés par des politiques. Dans la mesure où nous sommes intéressé par le comportement d'un système soumis à une politique de sécurité, nous nous focalisons sur les actions du système qui sont concernées par la politique. Lorsqu'une requête d'exécution d'une action est effectuée au sein d'un système dans un état donné, le nouvel état peut dépendre de la décision prise par la politique pour l'action en question. Par ailleurs, les réponses données par une politique à une requête d'action ne sont pas nécessairement binaires (accord/refus). Les actions que nous considérons sont alors des paires requêtes-décisions que nous appelerons dans la suite événement.

Une politique de sécurité dépend partiellement des informations qui caractérisent les états d'un système mais ne dépend pas de la façon dont évolue le système. Nous nous focalisons sur les informations qui sont pertinentes du point de vue de l'évolution du système régi par une politique de sécurité et nous regroupons ces informations sous la notion d'environnement, correspondant à la partie d'intérêt (pour les considérations de sécurité) des états d'un système. Les systèmes sont alors caractérisés par des transitions entre environnements et les politiques de sécurité sont spécifiées comme des processus de décision pour les requêtes d'exécution d'actions effectuées dans un environnement. Dans cette section, nous proposons dans un premier temps d'établir les définitions formelles d'environnement et d'événements et, partant de ces définitions, présentons les notions de systèmes et de politique de sécurité puis formalisons celle de l'application d'une politique à un système.

4.1.1 Théorie de sécurité

Pour pouvoir appliquer une politique sur un système, il est nécessaire que ces deux éléments s'expriment en fonction d'un vocabulaire commun permettant d'identifier les éléments auxquels ils font référence. Ce vocabulaire doit permettre de décrire les informations contenues dans chaque état ainsi que les événements (actions et décisions). La donnée d'un tel vocabulaire et de contraintes sémantiques sur les éléments de ce vocabulaire est appelée théorie de sécurité et se définit comme suit :

Définition 4.2. Une **théorie de sécurité** $\mathfrak{teo}$ est la donnée :
- d'une signature $\Sigma_{\mathfrak{teo}} = \{\mathcal{S}, \mathcal{F}\}$;
- d'une extension de $\Sigma_{\mathfrak{teo}}$, notée $\tilde{\Sigma}^{Env}_{\mathfrak{teo}} = (\mathcal{S}, \mathcal{F} \cup \mathcal{F}_{Env}, \mathcal{P}_{Env})$;
- d'une seconde extension de $\Sigma_{\mathfrak{teo}}$, notée $\tilde{\Sigma}^{Act}_{\mathfrak{teo}} = (\mathcal{S}, \mathcal{F}, \mathcal{P}_{Act})$ dont les éléments de $\mathcal{P}_{Act}$ sont appelés symboles d'actions,

- un ensemble fini $\mathcal{D}_{\mathfrak{teo}}$ de décisions,
- une application $\mathcal{C}_{\mathfrak{teo}}$ de $\mathcal{P}_{Env}$ dans $\{empty, total{-}order, partial{-}order\}$ telle que si $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}) \neq empty$, alors $\mathbf{p} : \mathsf{s} \times \mathsf{s}$ pour une certaine sorte $\mathsf{s} \in \mathcal{S}$ telle qu'il n'existe aucun symbole fonctionnel de coarité s ni de prédicat, hormis $\mathbf{p}$, dont le profil contient s (dans ce cas, s est dite abstraite).

Tous les symboles fonctionnels définis dans une théorie de sécurité sont considérés comme des constructeurs : ils ont donc une vocation descriptive et ne doivent pas modéliser de processus d'évaluation.

La signification intuitive de ces éléments est la suivante : la signature $\Sigma_{\mathfrak{teo}}$ définit les éléments communs aux états et aux transitions du système de sécurité que l'on modélise ; la signature $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$ décrit le langage des états tandis que $\tilde{\Sigma}^{Act}_{\mathfrak{teo}}$ définit le langage des étiquettes (c.-à-d. des actions).

Deux faits notables doivent être soulignés :

1. l'absence de symbole fonctionnel défini et
2. la présence de prédicats associés à des contraintes d'ordres et définis sur des sortes sans constructeur.

Ces deux aspects sont motivés par les techniques que nous utiliserons dans la section 4.3 pour permettre la vérification des systèmes et des politiques modélisés. Ces techniques reposent sur l'existence d'un isomorphisme entre l'ensemble des environnements construits sur une théorie de sécurité donnée et l'ensemble des termes construits sur une signature particulière (déterminée à partir de la théorie en question). Un environnement correspond à une interprétation de la signature $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$. Pour permettre le raisonnement automatique sur les environnements, les relations seront symboliquement représentées (*c.f.* détails dans la section 4.2) par des listes de tuples de termes. Ce type de modélisation ne permet pas d'assurer syntaxiquement le caractère fonctionnel d'une relation. Par exemple, si l'on considère un symbole défini $\mathbf{f} : \mathsf{s} \to \mathsf{s}'$, alors une interprétation de $\mathbf{f}$ sera représentée par un terme de la forme $\langle\, n_1, n'_1 \,\rangle :: \langle\, n_2, n'_2 \,\rangle :: \ldots$ indiquant que $\mathbf{f}$ est interprété par la fonction associant n'_1 à n_1, n'_2 à n_2, ... Cependant, le terme $\langle\, n_1, n'_1 \,\rangle :: \langle\, n_1, n''_1 \,\rangle :: \ldots$ ne correspondra à aucune interprétation de $\mathbf{f}$ puisqu'il représente une relation non fonctionnelle. On ne peut en effet pas assurer syntaxiquement (c.-à-d. par la seule donnée d'une signature) que tous les couples possèdent une première composante distincte. Pour autant, cela ne signifie pas que l'on ne peut pas effectuer d'association fonctionnelle entre un élément et un attribut. Il est évident que le cadre de spécification doit permettre d'associer, par exemple, un niveau de sécurité à tout utilisateur. La vision que l'on adopte dans notre cadre est fondée sur une représentation des éléments du système par l'ensemble des attributs qui le caractérisent. Un terme n'identifie pas un élément mais le décrit. La méthode proposée se distingue de la méthode de modélisation usuelle consistant à associer à tout élément un objet qui l'identifie (le plus souvent un entier ou une chaîne de caractères) et d'associer cet objet, au moyen de fonctions, aux attributs qui caractérisent l'élément qu'il représente. Suivant cette méthode, un individu pourrait être représenté par une chaîne correspondant à son nom. Pour lui associer une couleur de cheveux et une taille, on définirait alors deux fonctions *hair_color* et *height* à valeurs respectivement dans un ensemble fini de

couleurs et dans l'ensemble des entiers. Dans notre cadre, les éléments ne sont pas identifiés mais entièrement définis par les attributs qui leur sont associés. Ainsi, un individu au cheveux noirs mesurant 170cm pourrait être représenté par un terme de la forme $\mathbf{user}(\mathbf{black}, \mathbf{succ}^{170}(\mathbf{zero}))$. Il est alors important de noter qu'une telle modélisation implique que deux éléments possédant les mêmes caractéristiques sont indistinguables. En conséquence, lors du raisonnement, sont faites les hypothèses d'existence et d'unicité d'un élément correspondant à chaque terme. Si l'on souhaite modéliser le fait qu'il existe plusieurs éléments possédant les mêmes caractéristiques, il est alors nécessaire d'ajouter une composante au terme décrivant l'élément en question. Par exemple, si l'on souhaite indiquer qu'il existe plusieurs individus ayant une même couleur de cheveux et une même taille, alors ces derniers peuvent être modélisés par un terme de la forme $\mathbf{user}(n, \mathbf{black}, \mathbf{succ}^{170}(\mathbf{zero}))$ où n est un entier.

Le deuxième aspect remarquable de notre définition est la présence de prédicats associés à des contraintes d'ordres et définis sur des sortes sans constructeur. Nous l'avons évoqué précédemment, la représentation symbolique des relations ne permet pas de contraindre ces dernières par une théorie particulière. Toutefois, comme nous le verrons dans la section 4.2.1, il est possible de raisonner modulo l'existence d'un ordre total ou partiel sur les éléments du système. Ceci requiert en particulier de ne pas connaître la représentation explicite des éléments à comparer. Ceci peut paraître suprenant de prime abord et pourtant, si l'on analyse les principales politiques (et principaux modèles) dans la littérature, on s'aperçoit que les éléments qui sont ordonnés (comme par exemple les niveaux) n'ont pas d'intérêt autre que de permettre de comparer les éléments qui sont associés à ces niveaux. Par exemple, lorsque l'on attribue des niveaux de sécurité aux utilisateurs et aux objets, la question d'intérêt n'est pas de connaître le niveau d'un utilisateur particulier, mais de pouvoir le comparer avec le niveau d'un autre utilisateur ou d'un objet. Il est donc, dans ce contexte, inutile de donner un « nom » aux différents niveaux. Le raisonnement effectué sur les spécifications établies dans notre cadre est alors générique en ce sens où il n'est pas « attaché » à une instance particulière de certains éléments du système. En effet, dans la plupart des travaux existants, le raisonnement s'effectue après avoir fixé un ensemble de noms représentant les utilisateurs ou encore un ensemble de niveaux ainsi que l'ordre associé. Les questions que l'on se pose sont alors de la forme « étant donnés ces ensembles d'utilisateurs et de niveaux ainsi que cet ordre sur les niveaux, la politique vérifie-t-elle la propriété p ? ». Dans notre cadre, les questions d'intérêt sont de la forme « existe-t-il un ensemble d'utilisateurs et un ensemble de niveaux partiellement (ou totalement) ordonné pour lesquels la politique ne vérifie pas la propriété p ? ».

Pour faciliter l'écriture et la lecture des spécifications, nous nous munissons d'un langage dont la syntaxe est illustrée par l'exemple suivant.

> *Exemple 4.3.* Nous considérons dans cet exemple une théorie `teo` permettant de décrire un système de sécurité basé sur la notion de niveaux. Cette théorie décrit un système dans lequel les sujets (Subject) accèdent aux objets (Object) selon un mode d'accès (Mode). Chaque sujet et objet est partiellement caractérisé par un niveau de sécurité mais le mode de représentation des sujets et des objets n'importe pas (**user** : Nat × Level → User

et **object** : Nat × Level → Object). Les modes d'accès sont quant à eux fixés (**read**, **write** :→ Mode). Les niveaux sont partiellement ordonnés (**inf** : Level × Level ∈ $\mathcal{P}_{Env}$ et $\mathcal{C}_{\text{tco}}(\textbf{inf}) = \mathit{partial{-}order}$). Un état du système se caractérise par l'ordre partiel que nous venons d'évoquer ainsi que par l'ensemble des accès courants (**accesses** : Subject × Object × Mode), l'ensemble des sujets appartenant au groupe des « super-utilisateurs » (**sudo** : Subject) ainsi que les ensembles des utilisateurs sur listes rouge et noire (**redlist**, **blacklist** : Subject). Les actions possibles dans le système sont la prise et le relâchement d'un accès (**take**, **release** : Subject × Object × Mode). Enfin, les deux décisions considérées ici sont la permission et le refus ($\mathcal{D}_{\text{tco}} = \{\textbf{permit}, \textbf{deny}\}$).

SECURITY THEORY
Level_Based_Theory
Signature

$$\mathit{sorts} \triangleq \{\ \mathsf{Mode}, \mathsf{Subject}, \mathsf{Object}, \mathsf{Level}\ \}$$

$$\mathit{constructors} \triangleq \left\{ \begin{array}{rcl} \textbf{read} & : & \rightarrow \mathsf{Mode} \\ \textbf{write} & : & \rightarrow \mathsf{Mode} \end{array} \right\}$$

States

$$\mathit{constructors} \triangleq \left\{ \begin{array}{rcll} \textbf{user} & : & \mathsf{Nat} \times \mathsf{Level} & \rightarrow \mathsf{Subject} \\ \textbf{object} & : & \mathsf{Nat} \times \mathsf{Level} & \rightarrow \mathsf{Object} \\ \textbf{zero} & : & & \rightarrow \mathsf{Nat} \\ \textbf{succ} & : & \mathsf{Nat} & \rightarrow \mathsf{Nat} \end{array} \right\}$$

$$\mathit{predicates} \triangleq \left\{ \begin{array}{rcl} \textbf{sudo} & : & \mathsf{Subject} \\ \textbf{blacklist} & : & \mathsf{Subject} \\ \textbf{redlist} & : & \mathsf{Subject} \\ \textbf{inf}^{\text{p.o.}} & : & \mathsf{Level} \times \mathsf{Level} \\ \textbf{accesses} & : & \mathsf{Subject} \times \mathsf{Object} \times \mathsf{Mode} \end{array} \right\}$$

Events

$$\mathit{actions} \triangleq \left\{ \begin{array}{rcl} \textbf{take} & : & \mathsf{Subject} \times \mathsf{Object} \times \mathsf{Mode} \\ \textbf{release} & : & \mathsf{Subject} \times \mathsf{Object} \times \mathsf{Mode} \end{array} \right\}$$

$$\mathit{decisions} \triangleq \{\ \textbf{permit}, \textbf{deny}\ \}$$

Les contraintes définies par $\mathcal{C}_{\text{tco}}$ sont matérialisées par la mise en exposant de « p.o. » pour $\mathit{partial{-}order}$ et « t.o. » pour $\mathit{total{-}order}$.

4.1.2 Environnements et règles de transition

Un système de sécurité est un système de transitions décrivant la façon dont les informations de sécurité évoluent. Les états du système correspondent ainsi à une interprétation des prédicats fixés par la théorie sur laquelle il repose. Pour permettre une automatisation du raisonnement sur les politiques et les systèmes que l'on définit, nous considérerons une classe particulière d'interprétations, faute de quoi il serait impossible d'exploiter les spécifications pour effectuer des vérifications de façon automatique. Dans ce cadre, nous proposons de restreindre l'ensemble des interprétations considérées à celles interprétant les prédicats par des relations finies. Nous pensons que ceci ne limite en aucun cas l'intérêt de notre cadre dans la mesure où, de façon effective, les systèmes informatiques sont caractérisés par un ensemble fini de faits

(la mémoire d'un ordinateur étant, *de facto*, finie). En pratique, on s'aperçoit que les informations concernant le système sont stockées sous la forme d'une base de données finie (les bases de données étant composées de fichiers stockant les enregistrements). En revanche, si l'on considère que les ensembles de faits et d'informations sont finis, on considère également qu'ils peuvent être arbitrairement grands, autrement dit non bornés.

Définition 4.4. Soit $\mathfrak{teo}$ une théorie de sécurité. On appelle **environnement** sur $\mathfrak{teo}$ toute interprétation η de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$ telle que :

- pour tout prédicat $\mathbf{p}$ de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$ tel que $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}) = empty$, $\mathbf{p}_\eta$ est un ensemble fini,
- pour tout prédicat $\mathbf{p} : \mathsf{s} \times \mathsf{s}$ tel que $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}) = partial{-}order$,

$$\eta \models \begin{cases} \forall x : \mathsf{s},\ \mathbf{p}(x,x) \\ \forall x,y : \mathsf{s},\ \mathbf{p}(x,y) \wedge \mathbf{p}(y,x) \Rightarrow x = y \\ \forall x,y,z : \mathsf{s},\ \mathbf{p}(x,y) \wedge \mathbf{p}(y,z) \Rightarrow \mathbf{p}(x,z) \end{cases}$$

- pour tout prédicat $\mathbf{p} : \mathsf{s} \times \mathsf{s}$ tel que $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}) = total{-}order$,

$$\eta \models \begin{cases} \forall x : \mathsf{s},\ \mathbf{p}(x,x) \\ \forall x,y : \mathsf{s},\ \mathbf{p}(x,y) \wedge \mathbf{p}(y,x) \Rightarrow x = y \\ \forall x,y,z : \mathsf{s},\ \mathbf{p}(x,y) \wedge \mathbf{p}(y,z) \Rightarrow \mathbf{p}(x,z) \\ \forall x,y : \mathsf{s},\ \mathbf{p}(x,y) \vee \mathbf{p}(y,x) \end{cases}$$

On note $\mathcal{E}nv(\mathfrak{teo})$ l'ensemble des environnements sur $\mathfrak{teo}$.

Exemple 4.5. Si l'on considère la théorie de sécurité définie dans l'exemple précédent, on peut définir un environnement η de la façon suivante (seules les informations d'intérêt sont données, les autres pouvant être arbitraires ou entièrement déterminées par les informations présentées).

- $|\,\eta\,|_{\mathsf{Mode}} = \{read, write\}$;
- $|\,\eta\,|_{\mathsf{Subject}} = \{Alice, Bob, Charlie\}$;
- $|\,\eta\,|_{\mathsf{Object}} = \{f_1, f_2, f_3, f_4\}$;
- $|\,\eta\,|_{\mathsf{Level}} = \{public, defense, admin\}$;
- $\mathbf{read}_\eta = read$, $\mathbf{write}_\eta = write$
- $\mathbf{user}_\eta(1, admin) = Alice$, $\mathbf{user}_\eta(1, defense) = Bob$, $\mathbf{user}_\eta(2, defense) = Charlie$
- $\mathbf{object}_\eta(1, admin) = f_1$, $\mathbf{object}_\eta(1, defense) = f_2$, $\mathbf{object}_\eta(1, public) = f_3$, $\mathbf{object}_\eta(2, public) = f_4$
- $\mathbf{sudo}_\eta = \{Alice\}$
- $\mathbf{inf}_\eta = \{(public, defense), (public, admin)\}$
- $\mathbf{accesses}_\eta = \begin{cases} (Alice, f_1, read), (Alice, f_2, write) \\ (Bob, f_1, read), (Charlie, f_4, write) \end{cases}$
- $\mathbf{redlist}_\eta = \mathbf{blacklist}_\eta = \varnothing$

Rappelons que le rôle des environnements est d'établir la validité des faits à un instant donné de l'exécution du programme modélisé par le système. L'évolution

du système en fonction de l'éxecution d'une action, tout comme l'évaluation d'une requête d'action par une politique de sécurité, se définit en fonction de la validité de certaines contraintes dans l'environnement courant. On comprend alors que les environnements ont vocation à être « interrogés » par le système et les politiques à définir. L'un des principaux intérêts de considérer les interprétations de la forme évoquée est la possibilité de calculer les solutions d'une large de classe de formules dans un environnement donné (par solution on entend l'ensemble des valuations des variables libres rendant vraie la formule dans l'interprétation décrite par l'environnement). La famille de formules en question est bien connue de la communauté des bases de données, car, d'un point de vue théorique, une base de données est exactement une interprétation finie d'une signature particulière. Les formules considérées sont équivalentes aux requêtes de l'algèbre relationnelle [Abiteboul et al., 1995] et sont qualifiées de formules sûres. La définition usuelle des formules sûres [Ullman, 1988] est adaptée pour être compatible avec la notion de théorie de sécurité.

Définition 4.6. Soit $\mathfrak{teo}$ une théorie de sécurité. On appelle **formule sûre** sur $\mathfrak{teo}$ toute formule φ de $\mathcal{F}or^{=}(\tilde{\Sigma}^{Env}_{\mathfrak{teo}})$ vérifiant les conditions suivantes :

- (i) elle ne comporte aucun quantificateur universel (cette contrainte n'affecte pas l'expressivité de la classe de formules considérée puisque toute formule possède une forme équivalente vérifiant cette contrainte) ;
- (ii) pour toute disjonction $\varphi_1 \vee \varphi_2$ apparaissant dans φ, $\mathcal{FV}ar(\varphi_1) = \mathcal{FV}ar(\varphi_2)$;
- (iii) pour toute conjonction maximale (c.-à-d. n'étant pas en conjonction avec une autre sous-formule) $\varphi_1 \wedge \ldots \wedge \varphi_n$, les variables libres de chaque formule φ_i sont bornées dans le sens suivant :
  - une variable est bornée si elle apparaît dans un atome dont la racine est un prédicat $\mathbf{p}$ tel que $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}) = empty$ ou
  - si elle a une occurence dans un terme t apparaîssant dans une égalité de la forme $t = t'$ où t' est clos ou ne contient que des variables bornées ;
- (iv) toute sous-formule $\neg\varphi'$ est en conjonction avec une formule φ respectant les contraintes évoquées en (iii) et telle que $\mathcal{FV}ar(\varphi') \subseteq \mathcal{FV}ar(\varphi)$.

Par extension, nous appelerons formule sûre toute formule équivalente à une formule vérifiant les conditions citées.

Exemple 4.7. Soit $\mathfrak{teo}$ la théorie décrite dans l'exemple 4.3. La formule suivante :

$$\forall s, \forall o, \forall a, \left\{ \begin{array}{l} (\mathbf{accesses}(s,o,a) \wedge o = \mathbf{object}(i_o, l_o) \wedge \ \ s = \mathbf{user}(i_s, l_s)) \\ \quad \Rightarrow \mathbf{inf}(l_o, l_s) \end{array} \right\}$$

est une formule sûre sur $\mathfrak{teo}$. Elle est équivalente à la formule ψ, satisfaisant les conditions énoncées dans la définition 4.6, suivante :

$$\neg \left(\exists s, \exists o, \exists a, \left\{ \begin{array}{l} \mathbf{accesses}(s,o,a) \wedge o = \mathbf{object}(i_o, l_o) \wedge \ \ s = \mathbf{user}(i_s, l_s) \\ \quad \wedge \neg \mathbf{inf}(l_o, l_s) \end{array} \right\} \right)$$

En effet, l'atome $\mathbf{accesses}(s,o,a)$ borne les variables s, o et a pour la sous-formule ψ' :

$$\mathbf{accesses}(s,o,a) \wedge o = \mathbf{object}(i_o, l_o) \wedge \ \ s = \mathbf{user}(i_s, l_s) \wedge \neg \mathbf{inf}(l_o, l_s)$$

Les variables i_o, l_o, i_s et l_s sont bornées par les égalitées $o = \mathbf{object}(i_o, l_o)$ et $s = \mathbf{user}(i_s, l_s)$. L'atome $\mathbf{inf}(l_o, l_s)$ est donc sûr et par voie de conséquence le littéral $\neg\mathbf{inf}(l_o, l_s)$ l'est également (car en conjonction avec des sous-formules sûres). La formule ψ peut être considérée comme étant en conjonction avec $\top$. Comme $\mathcal{FV}ar(\psi) = \mathcal{FV}ar(\top)$, ψ' est sûre et $\top$ respecte les contraintes énoncées par le point (iii) de la définition, ψ est une formule sûre sur $\mathfrak{tco}$.

Les règles de transition d'un système de sécurité décrivent l'évolution des environnements en fonction d'une action donnée. Plusieurs approches ont été proposées dans la littérature pour spécifier des modifications de modèles logiques : un langage d'actions est présenté dans [Gelfond and Lifschitz, 1998] tandis que la notion de règles de mise à jour est proposée dans [Alferes et al., 2002]. L'approche que nous retenons est semblable, dans l'idée, à celle proposée par [Alferes et al., 2002].

Définition 4.8. Une règle de **mise à jour** sur une théorie de sécurité $\mathfrak{tco}$ est une règle de la forme $H \leftarrow \alpha$ ou $H \leftarrow \alpha \mid \beta$ où

- H est un littéral de $\tilde{\Sigma}^{Env}_{\mathfrak{tco}}$ dont le symbole de tête est un prédicat $\mathbf{p}$ tel que $\mathcal{C}_{\mathfrak{tco}}(\mathbf{p}) = empty$,
- α est une formule sûre de $\tilde{\Sigma}^{Env}_{\mathfrak{tco}}$,
- β est une conjonction d'atomes de la forme $\mathbf{p}(x, y)$, pour un certain $\mathbf{p}$ tel que $\mathcal{C}_{\mathfrak{tco}}(\mathbf{p}) \neq empty$, dont exactement une variable (c.-à-d. soit x, soit y) apparaît dans α.

La signification d'une règle de mise à jour est la suivante. Si la règle est de la forme $H \leftarrow \alpha$, alors elle indique l'ajout (si H est un littéral positif) ou le retrait (si H est un littéral négatif) des faits construits à partir de H à partir de toutes les valuations rendant vraie la formule α dans l'environnement courant. Si la règle est de la forme $H \leftarrow \alpha \mid \beta$, alors les conditions de β décrivent la mise à jour d'un ordre (total ou partiel) par l'ajout d'un nouvel élément dont les contraintes indiquent sa « position » dans l'ordre. Plus précisément, si l'on considère un ensemble totalement ordonné $(E, \leq)$, deux éléments $e_{before}, e_{after} \in E$ et $e \notin E$, alors la contrainte $e_{before} \leq e \leq e_{after}$ indique la mise à jour de $(E, \leq)$ par $(E \cup \{e\}, \leq')$ telle que : $\forall e_1, e_2 \in E,\ e_1 \leq' e_2$ ssi $e_1 \leq e_2$ et $\forall e_1, e_2 \in E,\ e_1 \leq' e$ ssi $e_1 \leq e_{before}$ et $e \leq' e_2$ ssi $e_{after} \leq e_2$ (ceci requiert que $e_1 \leq e_2$ et qu'il n'existe pas de e' tel que $e_{before} \leq e' \leq e_{after}$). De la même façon si l'on considère un ensemble partiellement ordonné $(E, \leq)$, deux ensembles finis $\{e^1_{before}, \ldots e^n_{before}\}$ et $\{e^1_{after}, \ldots e^m_{after}\}$ d'éléments de E et $e \notin E$, alors la contrainte $e^1_{before}, \ldots e^n_{before} \leq e \leq e^1_{after}, \ldots, e^m_{after}$ indique la mise à jour de $(E, \leq)$ par $(E \cup \{e\}, \leq')$ telle que : $\forall e_1, e_2 \in E,\ e_1 \leq' e_2$ ssi $e_1 \leq e_2$ et $\forall e_1, e_2 \in E,\ e_1 \leq' e$ ssi $\exists i, e_1 \leq e^i_{before}$ et $e \leq' e_2$ ssi $\exists j, e^j_{after} \leq e_2$ (ceci requiert que pour tout i, j, $e^i_{before} \leq e^j_{after}$). Pour qu'une contrainte ait un sens, il faut que chaque atome la composant associe un élément à « insérer » dans l'ordre (donc n'apparaissant pas dans α) à un élément « déjà existant » dans l'ordre (c.-à-d. apparaissant dans α).

Exemple 4.9. Nous considérons de nouveau la théorie décrite dans l'exemple 4.3. La règle :

$$\neg\mathbf{accesses}(s, o, a) \leftarrow \left\{ \begin{array}{l} \mathbf{accesses}(s, o, a) \wedge o = \mathbf{object}(i_o, l_o) \\ \wedge\ s = \mathbf{user}(i_s, l_s) \wedge \mathbf{inf}(l_o, l_s) \end{array} \right.$$

décrit la mise à jour consistant à supprimer tous les atomes représentant l'accès d'un objet par un sujet dont le niveau de sécurité est supérieur à celui de l'objet. Remarquez que dans la méthode de modélisation adoptée, la détermination du niveau de sécurité d'un sujet (ou objet) s'effectue par « filtrage », c.-à-d. en établissant une égalité entre la variable (qui sera instanciée au moment de l'application de la règle) et un motif dont les variables correspondent aux attributs que l'on souhaite déterminer. La règle :

$$\mathbf{sudo}(\mathbf{user}(\mathbf{succ}(id_3), l)) \leftarrow \begin{cases} t_1 = \mathbf{object}(id_1, l_1) \\ \wedge\ t_2 = \mathbf{object}(id_2, l_2) \\ \wedge\ t_3 = \mathbf{user}(id_3, l_3) \end{cases} \mid \begin{cases} \mathbf{inf}(l_1, l) \\ \wedge\ \mathbf{inf}(l_2, l) \\ \wedge\ \mathbf{inf}(l, l_2) \end{cases}$$

(les t_i sont des termes clos) décrit la mise à jour consistant à « créer » un nouvel utilisateur associé au groupe des super-utilisateurs et dont le niveau est un nouveau niveau situé entre les niveaux des objets décrits par t_1 et t_2 et le niveau de sécurité du sujet décrit par t_3.

L'application d'une mise à jour $u = H \leftarrow \alpha$ sur un environnement η, notée $\eta \oplus u$, correspond à l'ajout ou à la suppression des faits correspondant aux instances de H pour toute valuation valide de α. Formellement, ceci se décrit de la façon suivante, si l'on note η' l'environnement résultant de l'opération $\eta \oplus u$:

cas (i) $\mathbf{p}_{\eta'} = \mathbf{p}_\eta \cup \left(\bigcup_{\eta \models \mu(\alpha)} (\mu(t_1), \ldots, \mu(t_n))\right)$

cas (ii) $\mathbf{p}_{\eta'} = \mathbf{p}_\eta \setminus \left(\bigcup_{\eta \models \mu(\alpha)} (\mu(t_1), \ldots, \mu(t_n))\right)$

selon que (i) $H = \mathbf{p}(t_1, \ldots, t_n)$ ou (ii) $H = \neg\mathbf{p}(t_1, \ldots, t_n)$ et $\mathbf{q}_\eta = \mathbf{q}_{\eta'}$ pour tout $\mathbf{q} \neq \mathbf{p}$. L'application d'une mise à jour $u = H \leftarrow \alpha \mid \beta$ sur un environnement η s'effectue de la même façon que précédemment mais en considérant que la variable apparaîssant dans H et β mais n'apparaissant pas dans α est un nouvel élément à ajouter selon les remarques précisées antérieurement (remarques sur la mise à jour d'un ensemble ordonné).

L'application d'une séquence de mises à jour $U = (u_1, \ldots, u_n)$ sur un environnement η est l'environnement $\eta \oplus U = (\ldots(\eta \oplus u_1) \oplus u_2)\ldots) \oplus u_n)$.

Les transitions d'un système de sécurité décrivant l'évolution des états correspondent à des séquences de mises à jour étiquetées par des actions qui les « déclenchent ».

Définition 4.10. Soit $\mathfrak{teo}$ une théorie de sécurité. Une **règle de transition** sur $\mathfrak{teo}$ est une paire $event = (action, decision)$, appelée motif d'événement, associée à une séquence de mises à jour U sur $\mathfrak{teo}$ (dont les variables apparaissant dans $action$ sont considérées comme des constantes) et telle que $action$ est un atome de la forme $\mathbf{p}(x_1, \ldots, x_n)$ avec $\mathbf{p}$ symbole de prédicat de $\tilde{\Sigma}^{Act}_{\mathfrak{teo}}$ et $decision$ une décision de $\mathfrak{teo}$. Une règle de transition sera notée $(event) : U$. Toute règle de transition $r = (event) : U$ engendre une relation $\rightarrow_r$ définie de la façon suivante : $\eta \xrightarrow{evt}_r \eta'$ ssi $evt = \sigma(event)$ et $\eta' = \eta \oplus \sigma(U)$. De plus, pour tout ensemble de règles δ, $\eta \xrightarrow{evt}_\delta \eta'$ ssi il existe une règle $r \in \delta$ telle que $\eta \xrightarrow{evt}_r \eta'$.

Exemple 4.11. Nous définissons les règles de transition suivantes sur la théorie décrite dans l'exemple 4.3.

$$(i)\quad (\mathbf{take}(s,o,a), \mathbf{permit}) : \begin{cases} \mathbf{accesses}(s,o,a) & \leftarrow \neg\mathbf{accesses}(s,o,a) \\ \neg\mathbf{redlist}(s) & \leftarrow \\ \neg\mathbf{blacklist}(s) & \leftarrow \end{cases}$$

$$(ii)\quad (\mathbf{take}(s,o,a), \mathbf{deny}) : \begin{cases} \mathbf{blacklist}(s) & \leftarrow \mathbf{redlist}(s) \\ \neg\mathbf{accesses}(s,o',a') & \leftarrow \mathbf{blacklist}(s) \wedge \mathbf{accesses}(s,o',a') \\ \mathbf{redlist}(s) & \leftarrow \end{cases}$$

$$(iii)\quad (\mathbf{release}(s,o,a), \mathbf{permit}) : \{\ \neg\mathbf{accesses}(s,o,a) \ \leftarrow$$

Les règles (i) et (iii) indiquent que lorsqu'un accès est autorisé, les faits correspondant sont ajoutés et lorsqu'un accès est relâché, le fait est supprimé. La règle (ii) exprime le fait qu'un sujet est inscrit sur une liste rouge lorsqu'il tente d'effectuer un accès qui lui est interdit et qu'il est inscrit sur une liste noire lorsqu'il effectue deux tentatives successives. Lorsqu'un sujet est mis sur liste noire, tous ses accès courants sont supprimés.

Lorsque les événements des règles de transition ne sont pas unifiables, on obtient trivialement la propriété suivante :

Proposition 4.12. Pour tout ensemble δ de règles de transition disjointes (c'est-à-dire dont les motifs d'événements ne se superposent pas) sur une théorie $\mathfrak{teo}$, la relation $\rightarrow_\delta$ est déterministe.

4.1.3 Système de sécurité

À partir des définitions d'environnements et de règles de transition on peut désormais définir la notion de système de sécurité. La caractérisation des environnements ainsi que les transitions du système s'effectuent modulo une relation d'équivalence entre ces derniers.

Définition 4.13. Soit $\mathfrak{teo}$ une théorie de sécurité. On définit la **relation d'équivalence** $\approx_{\mathfrak{teo}}$ sur $\mathcal{E}nv(\mathfrak{teo})$ de la façon suivante : $\eta \approx_{\mathfrak{teo}} \eta'$ ssi il existe deux ensembles $E \subseteq |\eta|$ et $E' \subseteq |\eta'|$ en bijection par un certain isomorphisme Φ tels que pour tout prédicat $\mathbf{p} \cup \{=\}$ de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$, $(t_1,\ldots,t_n) \in \mathbf{p}_\eta$ ssi $(\Phi(t_1),\ldots,\Phi(t_n)) \in \mathbf{p}_{\eta'}$.

On s'aperçoit que si de tels E et E' existent, alors ils contiennent nécessairement l'ensemble des éléments du domaine d'interprétation (respectivement $|\eta|$ et $|\eta'|$) ayant au moins une occurence dans une interprétation d'un prédicat. Si l'on note E_{min} et E'_{min} les ensembles correspondants (les « domaines d'interprétation effectifs »), alors $\eta \approx_{\mathfrak{teo}} \eta'$ ssi il existe un isomorphisme Φ entre E_{min} et E'_{min} tels que pour tout prédicat $\mathbf{p} \cup \{=\}$ de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$, $(t_1,\ldots,t_n) \in \mathbf{p}_\eta$ ssi $(\Phi(t_1),\ldots,\Phi(t_n)) \in \mathbf{p}_{\eta'}$.

Définition 4.14. Un **système de sécurité** $\mathfrak{S}$ sur une théorie $\mathfrak{teo}$ est la donnée d'un ensemble de règles de transition sur $\mathfrak{teo}$ ainsi que d'une formule ϕ_{init} sans variable libre de la forme $\exists x_1, \ldots, x_n, \psi$ où ψ est une conjonction d'atomes et $x_1, \ldots, x_n$ sont des variables d'une sorte abstraite. Un système de sécurité $\mathfrak{S} = (\phi_{init}, \Delta)$ engendre un ensemble de systèmes de transitions isomorphes dont les états sont des environnements, l'état initial est un environnement correspondant à un modèle minimal de ϕ_{init} et les transitions sont des événements dont les actions sont construites sur $\mathcal{P}_{Act}$ et $|\,\eta\,|$, c.-à-d. des termes de la forme $\mathbf{act}(e_1, \ldots, e_n)$ où $\mathbf{act} : \mathsf{s_1} \times \ldots \times \mathsf{s_n} \in \mathcal{P}_{Act}$ et $e_i \in |\,\eta\,|_{\mathsf{s_i}}$ pour tout i.

Exemple 4.15. En reprenant la théorie de sécurité définie dans les exemples précédents, on définit le système de sécurité suivant :

SECURITY SYSTEM
Level_Based_System
Theory
Level_Based_Theory
Events
$(\mathbf{take}(s, o, a), \mathbf{permit})$:

$$\begin{cases} \mathbf{accesses}(s,o,a) & \leftarrow & \neg\mathbf{accesses}(s,o,a) \\ \neg\mathbf{redlist}(s) & \leftarrow & \\ \neg\mathbf{blacklist}(s) & \leftarrow & \end{cases}$$

$(\mathbf{take}(s, o, a), \mathbf{deny})$:

$$\begin{cases} \mathbf{blacklist}(s) & \leftarrow & \mathbf{redlist}(s) \\ \neg\mathbf{accesses}(s,o',a') & \leftarrow & \begin{cases} \mathbf{blacklist}(s) \\ \wedge\, \mathbf{accesses}(s,o',a') \end{cases} \\ \mathbf{redlist}(s) & \leftarrow & \end{cases}$$

$(\mathbf{release}(s, o, a), \mathbf{permit})$:

$$\{\ \neg\mathbf{accesses}(s,o,a) \quad \leftarrow$$

Initial environnement
$\exists l{:}\mathsf{Level}, \mathbf{sudo}(\mathbf{user}(\mathbf{zero}, l))$

La formule décrivant la situation initiale du système indique, dans cet exemple, l'existence d'un individu appartenant au groupe des super-utilisateurs.
L'environnement η décrit dans l'exemple 4.5 est un environnement initial du système. De plus, pour $act = \mathbf{take}(Alice, f_1, write)$, on a $\eta \xrightarrow{(act, \mathbf{permit})}_{\mathfrak{S}} \eta'$ avec $|\,\eta'\,| = |\,\eta\,|$, $\mathbf{p}_{\eta'} = \mathbf{p}_\eta$ pour tout $\mathbf{p} \neq \mathbf{accesses}$ et $\mathbf{accesses}_{\eta'} = \mathbf{accesses}_\eta \cup \{(Alice, f_1, write)\}$.

Il est important de noter que le choix du représentant d'une classe d'équivalence (selon $\approx_{\mathfrak{teo}}$) d'environnements n'affecte pas le calcul des états obtenus après transition. La relation d'équivalence $\approx_{\mathfrak{teo}}$ est en effet caractérisée par la propriété essentielle suivante :

Proposition 4.16. Soient $\mathfrak{teo}$ une théorie de sécurité et η et η' deux environnements sur $\mathfrak{teo}$ équivalent pour $\approx_{\mathfrak{teo}}$. On a la propriété suivante : $\eta \models \varphi$ ssi $\eta' \models \varphi$

pour toute formule sûre φ de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$.

❝ *Démonstration.* Puisque $\eta \approx_{\mathfrak{teo}} \eta'$, il existe un isomorphisme Φ entre les domaines effectifs de η et η'. En conséquence, à toute η-valuation correspond exactement une η'-valuation et réciproquement. On obtient alors l'équivalence de la validité d'une formule indépendante des domaines d'interprétations pour η et η'. La sémantique d'une formule sûre ne dépendant pas du domaine d'interprétation considéré mais uniquement de l'interprétation des prédicats, $\eta \models \varphi$ ssi $\eta' \models \varphi$ pour toute formule sûre φ de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$. ❞

4.1.4 Politiques de sécurité

Comme nous l'avons déjà évoqué, une politique est un processus évaluant les requêtes d'actions en une décision compte-tenu de l'état courant du système (l'environnement courant). Nous nous proposons de décrire ce processus au moyen de règles de réécriture contraintes. L'idée consistant à utiliser des règles de réécriture contraintes pour spécifier des politiques de sécurité a été initiée dans [Bourdier et al., 2010b, Bourdier et al., 2011]. L'utilisation combinée des règles de réécriture, des contraintes du premier ordre et d'une stratégie d'évaluation nous permet d'écrire des spécifications de façon extrêmement naturelle et aisée. Le pouvoir d'expression offert par le formalisme proposé permet de décrire des politiques complexes dans un langage simple.

Définition 4.17. Soit $\mathfrak{teo}$ une théorie de sécurité. Une **politique de sécurité** $\mathfrak{pol}$ sur $\mathfrak{teo}$ est un ensemble de règles de réécriture de la forme suivante :

$$lhs \xrightarrow{\varphi} rhs \quad (\textit{aussi écrite } lhs \to rhs \parallel \varphi)$$

où φ est une formule sûre de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$ et lhs et rhs sont des atomes de $\tilde{\Sigma}^{Act}_{\mathfrak{teo}}$ (vus comme des termes), ou (pour rhs seulement) une décision (vue comme une constante) ainsi qu'une stratégie de choix de règles exprimée sous la forme d'un ordre partiel sur les règles noté $<_{\mathfrak{pol}}$. À toute politique $\mathfrak{pol}$ est associée la relation de réduction suivante (définie relativement à un environnement η) : $t \to^{\eta}_{\mathfrak{pol}} t'$ si et seulement si il existe une règle de réécriture $r = lhs \xrightarrow{\varphi} rhs$ et une substitution close σ telle que $t = \sigma(lhs)$, $t' = \sigma(rhs)$ et $\eta \models \sigma(\varphi)$ et telle qu'il n'existe aucune règle $r' = lhs' \xrightarrow{\varphi'} rhs' <_{\mathfrak{pol}} r$ telle que $t = \sigma'(lhs')$ et $\eta \models \sigma'(\varphi')$ pour une certaine substitution close σ'.

Exemple 4.18. La spécification suivante :

SECURITY POLICY
Level_Based_Policy
Theory
Level_Based_Theory

Rules

[1] $\mathbf{take}(s, o, a) \rightarrow \mathbf{deny} \quad \| \quad \mathbf{blacklist}(s)$

[2] $\mathbf{take}(s, o, a) \rightarrow \mathbf{permit} \quad \| \quad \mathbf{sudo}(s)$

[3] $\mathbf{take}(s, o, \mathbf{read}) \rightarrow \mathbf{permit} \quad \|$

$$\begin{cases} s = \mathbf{user}(i_s, l_s) \\ \wedge\ o = \mathbf{object}(i_o, l_o) \\ \wedge\ \mathbf{inf}(l_o, l_s) \\ \wedge\ (\forall o', \mathbf{accesses}(s, o', \mathbf{write}) \\ \quad \wedge\ o' = \mathbf{object}(i_{o'}, l_{o'}) \\ \quad \Rightarrow\ \mathbf{inf}(l_o, l_{o'})) \end{cases}$$

[4] $\mathbf{take}(s, o, \mathbf{write}) \rightarrow \mathbf{permit} \quad \|$

$$\begin{cases} s = \mathbf{user}(i_o, l_o) \\ \wedge\ (\forall o', \mathbf{accesses}(s, o', \mathbf{read}) \\ \quad \wedge\ o' = \mathbf{object}(i_{o'}, l_{o'}) \\ \quad \Rightarrow\ \mathbf{inf}(l_{o'}, l_o)) \end{cases}$$

[5] $\mathbf{take}(s, o, a) \rightarrow \mathbf{deny}$

[6] $\mathbf{release}(s, o, a) \rightarrow \mathbf{permit}$

Strategy

$1 < 2$

$2 < 3, 4$

$3 < 5$

$4 < 5$

décrit une politique indiquant que :

- un sujet sur liste noire ne peut obtenir aucun accès,
- un sujet appartenant au groupe des super-utilisateurs, s'il n'est pas sur liste noire, peut obtenir n'importe quel accès,
- un sujet peut lire un objet dont le niveau de sécurité est plus petit que son niveau s'il n'a pas d'accès en écriture sur un object de niveau inférieur,
- un sujet peut obtenir un accès en écriture sur un objet s'il n'a pas accès en lecture à un objet dont de niveau supérieur,
- un accès peut être relâché sans condition.

Enfin, on définit les propriétés suivantes sur les politiques de sécurité.

Définition 4.19. Soit $\mathfrak{teo}$ une théorie de sécurité. Une politique de sécurité $\mathfrak{pol}$ sur $\mathfrak{teo}$ est dite :

- **cohérente** ssi pour tout environnement η et toute action act, il existe au plus une décision d telle que $act \xrightarrow{*}{}^{\eta}_{\mathfrak{pol}} d$;
- **terminante** ssi pour tout environnement η, la relation $\xrightarrow{*}{}^{\eta}_{\mathfrak{pol}}$ termine ;
- **complète** ssi pour tout environnement η et toute action act, il existe au moins une décision d telle que $act \xrightarrow{*}{}^{\eta}_{\mathfrak{pol}} d$.

Comme évoqué dans le chapitre précédent, la notion de cohérence est un problème récurrent dans les approches à base de règles. En effet, les spécifications de politiques à partir de règles peuvent facilement engendrer des calculs ambigus. La notion de stratégie a été introduite dans notre cadre pour permettre de réduire aisément l'indéterminisme des réductions, facilitant ainsi la description de politiques

cohérentes.

La terminaison est également une question essentielle dans les langages de spécification qui permettent de définir des appels récursifs. Certains langages, comme Datalog, permettent seulement la définition de politiques terminantes, mais au prix d'une restriction importante du pouvoir d'expression.

La complétude est quant à elle un sujet symptomatique des approches effectuant un raisonnement « par cas ». On appelle parfois cette propriété « totalité » car similaire à la notion du même nom en théorie des relations. Certains formalismes « forcent » cette propriété en permettant la définition de « cas par défaut ». Nous proposons dans notre cadre d'utiliser une combinaison des stratégies et de règles par défaut pour faciliter l'écriture de politiques complètes.

4.1.5 Système sécurisé

L'objectif global dans lequel s'inscrivent les définitions de politiques et de systèmes de sécurité est l'élaboration d'un système sécurisé, c'est-à-dire un système répondant aux exigences de sécurité décrites par une politique. Nous définissons ainsi l'objet résultant de la combinaison d'un système de sécurité et d'une politique, ou, plus précisément, de l'application d'une politique de sécurité sur un système. L'objet obtenu est appelé système sécurisé.

Définition 4.20. Soient $\mathfrak{teo}$ une théorie de sécurité, $\mathfrak{S}$ un système de sécurité sur $\mathfrak{teo}$ et $\mathfrak{pol}$ une politique de sécurité terminante, cohérente et complète sur $\mathfrak{teo}$. L'application de $\mathfrak{pol}$ sur $\mathfrak{S}$ induit une famille de systèmes de transitions, appelés **systèmes $\mathfrak{pol}$-sécurisés obtenus à partir de $\mathfrak{S}$**, équivalents à un isomorphisme près, dont les états initiaux sont ceux des systèmes de transitions correspondants engendrés par $\mathfrak{S}$ et dont les transitions sont données par : $\eta \xrightarrow{act}_{\mathfrak{S}_{\mathfrak{pol}}} \eta'$ si et seulement si $\eta \xrightarrow{(act,d)}_{\mathfrak{S}} \eta'$ et $act \xrightarrow{*}^{\eta}_{\mathfrak{pol}} d$.

Dans la mesure où les systèmes sécurisés sont définis sous l'hypothèse de cohérence, complétude et de terminaison, la relation engendrée par un système sécurisé est calculable.

4.2 Sémantiques basées sur la réécriture

Nous avons proposé dans la section précédente des langages basés sur différents formalismes permettant de faciliter la spécification des différentes parties qui composent la modélisation d'un système sécurisé (règles de mises à jour, formules du premier ordre, règles de réécriture contraintes). Nous nous proposons dans cette section de donner une sémantique unifiée à chacun de ces composants (système et politique) ainsi qu'à leur combinaison (application d'une politique sur un système) basée sur des systèmes de réécriture particuliers afin d'en permettre une analyse automatique.

4.2.1 Représentation symbolique des environnements

Pour permettre une exploitation des spécifications exprimées dans notre cadre par des outils de vérification automatique, nous proposons de représenter chacun des objets de notre étude au moyen de termes algébriques et de systèmes de réécriture. La première notion dont nous allons discuter de la représentation symbolique est celle d'enrivonnement. Nous avons dans la précédente section proposé une définition des environnements sous la forme de modèles logiques du premier ordre interprétant les prédicats finitairement et validant des contraintes imposant l'interprétation de certains prédicats par des ordres (totaux ou partiels). Les contraintes imposées permettent de construire une correspondance entre l'ensemble des environnements d'une théorie de sécurité donnée et l'ensemble des termes clos d'une signature particulière.

On pourrait penser de premier abord que les contraintes imposées sont artificielles et que l'on peut naturellement représenter n'importe quel modèle fini par un terme sur une signature appropriée. Pourtant, s'il est possible d'associer à toute interprétation finie un terme algébrique, il se peut qu'un terme ne représente pas un modèle d'une théorie donnée. En effet, considérons une signature Σ munie d'une théorie $\mathcal{T}h$ et admettons que l'on construise une application qui à toute interprétation I finie de Σ associe un terme $t_I \in \mathcal{T}(\Sigma^{sym})$ (pour une certaine signature Σ^{sym}) et une autre application qui à une propriété α sur $\mathcal{M}od(\mathcal{T}h)$ associe une propriété p_α sur les termes $\mathcal{T}(\Sigma^{sym})$. Si l'on souhaite effectuer une preuve de la forme « pour tout $I \in \mathcal{M}od(\mathcal{T}h)$, $\alpha(I)$ » en vérifiant que « pour tout $t_I \in \mathcal{T}(\Sigma^{sym}), p_\alpha(t_I)$ » et qu'il existe des termes de $\mathcal{T}(\Sigma^{sym})$ représentant des interprétations de Σ qui ne sont pas modèles de $\mathcal{T}h$, alors la preuve de « pour tout $t_I \in \mathcal{T}(\Sigma^{sym}), p_\alpha(t_I)$ » peut échouer alors que la propriété α est vérifiée pour tous les modèles de $\mathcal{M}od(\mathcal{T}h)$.

Déterminer une représentation symbolique de l'ensemble des environnements d'une théorie donnée, c'est déterminer une signature dont chaque terme représente un environnement. C'est l'objet de la proposition suivante.

Proposition 4.21. Pour toute théorie de sécurité $\mathfrak{teo}$, il existe :
- une signature $\Sigma^{sym}_{\mathfrak{teo}}$,
- une surjection $[\![\cdot]\!]_{\mathfrak{teo}}$ de $\mathcal{T}(\Sigma^{sym}_{\mathfrak{teo}})$ sur $\mathcal{E}nv(\mathfrak{teo})_{/\approx_{\mathfrak{teo}}}$ et
- une injection $\gamma_{\mathfrak{teo}}$ de $\mathcal{E}nv(\mathfrak{teo})_{/\approx_{\mathfrak{teo}}}$ dans $\mathcal{T}(\Sigma^{sym}_{\mathfrak{teo}})$

telles que pour tout environnement $\eta \in \mathcal{E}nv(\mathfrak{teo})$, $[\![\gamma_{\mathfrak{teo}}(\eta)]\!] \approx_{\mathfrak{teo}} \eta$.

La construction de la signature $\Sigma^{sym}_{\mathfrak{teo}}$ est assez intuitive. On se base sur l'idée très classique selon laquelle à toute classe d'équivalence de $\mathcal{E}nv(\mathfrak{teo})_{/\approx_{\mathfrak{teo}}}$ peut être associé un représentant sous la forme d'une interprétation de Herbrand (sous l'hypothèse que le domaine d'interprétation des sortes abstraites est donné par un ensemble dénombrable de constantes quelconques). De plus, le caractère fini des interprétations des prédicats nous permet de représenter naturellement l'ensemble des atomes clos valides dans l'environnement considéré sous la forme d'une liste, les atomes clos construits sur la base de Herbrand de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$ étant vus comme des termes. On dispose alors d'une représentation naturelle des interprétations finies de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$. Dès lors, la question qui se pose est de savoir comment matérialiser des contraintes sémantiques. Les contraintes spécifient que l'interprétation de chaque prédicat est soit arbitraire

(ensemble vide de contraintes), soit correspond à un ordre total, soit correspond à un ordre partiel. Dans le premier cas, la représentation sous la forme d'une liste de faits convient. Dans les deux autres cas, il est nécessaire de modéliser autrement l'interprétation des prédicats concernés.

Considérons dans un premier temps un prédicat dont l'interprétation doit être un ordre total. Notons $\mathbf{inf} : \mathsf{s} \times \mathsf{s}$ le symbole correspondant. Pour matérialiser le fait que $\mathbf{inf}_\eta$ est un ordre total sur $|\,\eta\,|_{\mathsf{s}}$, nous nous fondons sur le théorème suivant :

Théorème 4.22. Tout ensemble fini totalement ordonné est isomorphe pour l'ordre à un segment initial de $\mathbb{N}$.

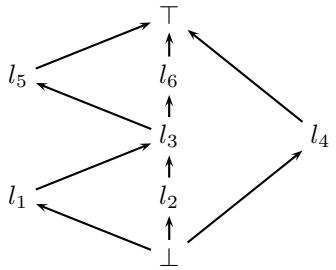
En conséquence, il suffit de représenter symboliquement les termes de sorte s par des termes construits sur :

$$\begin{cases} \mathbf{zero} & : & & \to & \mathsf{Nat} \\ \mathbf{succ} & : & \mathsf{Nat} & \to & \mathsf{Nat} \end{cases}$$

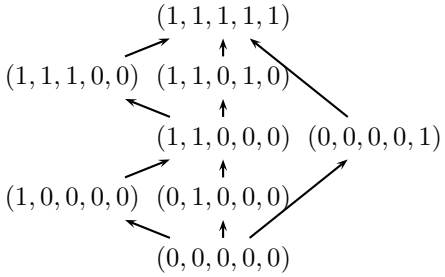
pour pouvoir déterminer de façon unique l'interprétation de $\mathbf{inf}_\eta$. De la même façon, la propriété suivante nous permet de représenter symboliquement les ordres partiels :

Théorème 4.23. Tout ensemble fini partiellement ordonné est isomorphe pour l'ordre à un sous-ensemble de $\mathbb{B}^n$, pour un certain $n \in \mathbb{N}$, muni de l'ordre produit (noté $<_\times$).

Exemple 4.24. Considérons un ensemble $E = \{\bot, l_1, \ldots, l_6, \top\}$ partiellement ordonné dont l'ordre $\leq_E$ est décrit par la figure suivante :



On peut construire un isomorphisme de $(E, \leq_E)$ sur un sous-ensemble de $(\mathbb{B}^5, \leq_\times)$. L'image de $(E, \leq_E)$ est illustré par la figure suivante :



Il suffit donc dans ce cas de représenter symboliquement les éléments de sorte s par un terme décrivant une liste de booléens :

$$\left\{\begin{array}{rcll} \mathbf{0} & : & \mathsf{BoolList} & \rightarrow \mathsf{BoolList} \\ \mathbf{1} & : & \mathsf{BoolList} & \rightarrow \mathsf{BoolList} \\ \mathbf{nil} & : & & \rightarrow \mathsf{BoolList} \end{array}\right.$$

À partir de ces remarques, nous pouvons montrer la proposition 4.21.

❝ *Démonstration.* Soit $\mathfrak{teo}$ une théorie de sécurité. La signature $\Sigma^{Sym}_{\mathfrak{teo}} = (\mathcal{S}, \mathcal{F})$ est définie comme suit :

- $\mathcal{S}$ contient toutes les sortes s de $\Sigma_{\mathfrak{teo}}$ ainsi que les sortes Nat, $\mathsf{BoolList}$, $\mathsf{Environment}$ et $\mathsf{Fact_p} < \mathsf{Fact}$ et $\mathsf{Facts_p} < \mathsf{Facts}$ pour tout prédicat $\mathbf{p}$ de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$.
- $\mathcal{F}$ contient tous les symboles suivants :
  - pour tout $\mathbf{f} : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \rightarrow \mathsf{s}$ de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$, le symbole $\mathbf{f} : \tau(\mathsf{s}_1) \times \ldots \times \tau(\mathsf{s}_n) \rightarrow \mathsf{s}$ de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$, où $\tau(\mathsf{s}_i)$ correspond à :
    - s_i si s_i n'est pas abstraite ;
    - $\mathsf{BoolList}$ si s_i est abstraite et s'il existe un prédicat $\mathbf{p} : \mathsf{s} \times \mathsf{s}$ tel que $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}) = \mathit{partial{-}order}$;
    - Nat si s_i est abstraite et s'il existe un prédicat $\mathbf{p} : \mathsf{s} \times \mathsf{s}$ tel que $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}) = \mathit{total{-}order}$;
  - pour tout prédicat $\mathbf{p} : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n$ tel que $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}) = \mathit{empty}$, les symboles :
    - $\mathbf{p} : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \rightarrow \mathsf{Fact_p}$,
    - $\mathbf{nil_p} : \rightarrow \mathsf{Facts_p}$
    - $\mathbf{cons_p} : \mathsf{Fact_p} \times \mathsf{Facts_p} \rightarrow \mathsf{Facts_p}$
  - le symbole $\langle _, \ldots, _ \rangle : \mathsf{Facts_{p_1}} \times \ldots \times \mathsf{Facts_{p_n}} \rightarrow \mathsf{Environment}$ où $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ est l'ensemble des prédicats de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$ tels que $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}) = \mathit{empty}$.

Soit un environnement $\eta \in \mathcal{E}nv(\mathfrak{teo})$. Notons $\tilde{\eta}$ l'environnement équivalent à η correspondant à une interprétation de Herbrand de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}}$. Pour tout $\mathbf{p}$ tel que $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}) = \mathit{empty}$, on construit le terme $\gamma_{\mathfrak{teo}}(\mathbf{p})_\eta$ comme étant la liste (construite avec les constructeurs $\mathbf{cons_p}$ et $\mathbf{nil_p}$ des termes $\mathbf{p}(\tau(t_1), \ldots, \tau(t_n))$, pour tout fait $\mathbf{p}(t_1, \ldots, t_n)$ valide dans $\tilde{\eta}$, où $\tau(t_i)$ est construit à partir de t_i en remplaçant les sous-termes de sorte abstraite par un élément de Nat ou de $\mathsf{BoolList}$ (selon que le terme modélise un ensemble totalement ou partiellement ordonné) de telle sorte que les ordres (et les égalités) sur les $\tau(t_i)$ respectent les ordres (et les égalités) de η. Le terme $\gamma_{\mathfrak{teo}}(\eta)$ est alors égal à $\langle \gamma_{\mathfrak{teo}}(\mathbf{p}_1)_\eta, \ldots, \gamma_{\mathfrak{teo}}(\mathbf{p}_n)_\eta \rangle$ (on ordonne arbitrairement les prédicats). Enfin, l'application $[\![\cdot]\!]_{\mathfrak{teo}}$ s'obtient simplement en effectuant le raisonnement inverse. ❞

> *Exemple 4.25.* Soit η l'environnement défini dans l'exemple 4.5. $\gamma_{\mathfrak{teo}}(\eta)$ est égal au terme $\langle t_{sudo}, t_{accesses}, t_{redlist}, t_{blacklist} \rangle$ où t_{sudo} est donné par (pour plus de lisibilité, les termes spécifiant les entiers seront représentés par les entiers correspondants)
>
> $$\mathbf{cons_{sudo}}(\mathbf{sudo}(\mathbf{user}(1, \mathbf{level}(\mathbf{0}(\mathbf{1}(\mathbf{nil}))))), \mathbf{nil_{sudo}})$$

$t_{accesses}$ par (les symboles $\mathbf{cons_{accesses}}$ et $\mathbf{nil_{accesses}}$ sont omis pour des questions de lisibilités) :

$$\begin{cases} \mathbf{accesses}(\mathbf{user}(1, \mathbf{level}(\mathbf{0}(\mathbf{1}(\mathbf{nil})))), \mathbf{object}(1, \mathbf{level}(\mathbf{0}(\mathbf{1}(\mathbf{nil})))), \mathbf{read}), \\ \mathbf{accesses}(\mathbf{user}(1, \mathbf{level}(\mathbf{0}(\mathbf{1}(\mathbf{nil})))), \mathbf{object}(1, \mathbf{level}(\mathbf{1}(\mathbf{0}(\mathbf{nil})))), \mathbf{write}), \\ \mathbf{accesses}(\mathbf{user}(1, \mathbf{level}(\mathbf{1}(\mathbf{0}(\mathbf{nil})))), \mathbf{object}(1, \mathbf{level}(\mathbf{0}(\mathbf{1}(\mathbf{nil})))), \mathbf{read}), \\ \mathbf{accesses}(\mathbf{user}(2, \mathbf{level}(\mathbf{1}(\mathbf{0}(\mathbf{nil})))), \mathbf{object}(2, \mathbf{level}(\mathbf{0}(\mathbf{0}(\mathbf{nil})))), \mathbf{write}) \end{cases}$$

$t_{redlist} = \mathbf{nil_{redlist}}$ et $t_{blacklist} = \mathbf{nil_{blacklist}}$.

4.2.2 Évaluation des contraintes

Pour permettre la spécification des règles de transition et de politiques de sécurité, il est nécessaire de pouvoir évaluer les formules sûres dans un environnement donné à partir de sa représentation symbolique. La méthode retenue doit permettre de vérifier les propriétés précédemment évoquées sur les systèmes de réécriture ainsi que l'analyse des systèmes sécurisés obtenus. Les développements récents concernant la vérification de propriétés de systèmes décrits par des systèmes de réécriture ([Rusu, 2010, Rocha and Meseguer, 2011, Rocha, 2011]) nécessitent l'introduction de la notion de systèmes de réécriture « topmost » ou système de réécriture « à la racine ».

Définition 4.26. Soit Σ une signature. On appelle **système de réécriture à la racine** sur Σ tout système de réécriture $\mathcal{R}$ tel qu'il existe une sorte $\mathsf{s_{top}}$ de Σ vérifiant les propriétés suivantes :
- pour tout symbole $\mathbf{f} : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \to \mathsf{s}$, $\mathsf{s}_i \neq \mathsf{s_{top}}$,
- pour toute règle $lhs \to rhs$ de $\mathcal{R}$, lhs et rhs sont des termes de sorte $\mathsf{s_{top}}$.

Les travaux évoqués présentent des méthodes de vérification de propriétés pour les systèmes de réécriture à la racine. C'est la raison pour laquelle nous nous attachons dans la suite à montrer que les différents processus (évaluation des requêtes, relations de transition, politiques de sécurité, ...) peuvent être modélisés au moyen de tels systèmes.

Proposition 4.27. Soit $\mathfrak{teo}$ une théorie de sécurité, il existe :
- une extension de $\Sigma^{sym}_{\mathfrak{teo}}$ notée $\Sigma^{AR}_{\mathfrak{teo}}$,
- un système de réécriture à la racine $\mathcal{R}_{AR}$ sur $\Sigma^{AR}_{\mathfrak{teo}}$, convergent,

tels que pour toute formule sûre φ de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$, il existe :
- un terme $\gamma_{\mathfrak{teo}}(\varphi)$ de $\mathcal{T}(\Sigma^{AR}_{\mathfrak{teo}}, \mathcal{X})$,
- une application $\sigma_\varphi : \mathcal{V}ar(\gamma_{\mathfrak{teo}}(\varphi)) \to \mathcal{P}$

tels que pour tout environnement $\eta \in \mathcal{E}nv(\mathfrak{teo})$:

$$\sigma^\eta_\varphi(\gamma_{\mathfrak{teo}}(\varphi)) \xrightarrow{*}_{\mathcal{R}_{AR}} \mathbf{done}(\langle\, t^1_1, \ldots, t^1_n \,\rangle :: \ldots :: \langle\, t^m_1, \ldots, t^m_n \,\rangle :: [\,])$$

si et seulement si $\eta \models \{x_1 \mapsto t^k_1, \ldots, x_n \mapsto t^k_n\}(\varphi)$ pour tout k (à l'ordre des éléments de la liste près) où σ^η_φ associe à $x \in \mathcal{V}ar(\gamma_{\mathfrak{teo}}(\varphi))$ le terme $\gamma_{\mathfrak{teo}}(\sigma_\varphi(x))_\eta$

(**done** est un symbole de Σ_{tco}^{AR} et :: et [] sont utilisés en remplacement de **cons** et **nil** pour des raisons de lisibilité).

❝ *Démonstration.* Soit φ une formule sûre sur $\tilde{\Sigma} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$. Alors, le théorème de Codd nous assure que l'on peut construire une expression basée sur les opérateurs de l'algèbre relationnelle évaluant φ dans n'importe quelle interprétation dans laquelle les prédicats sont interprétés par des relations finies [Codd, 1972] (l'algorithme correspondant est connu sous le nom d'algorithme de réduction de Codd). En conséquence, étant donnée une formule sûre φ, on peut calculer une expression de l'algèbre relationnelle exprimée en fonction des prédicats de $\mathcal{P}$. Par exemple, la formule $\varphi = \forall x, \mathbf{q}(x) \Rightarrow \exists y, \mathbf{p}(x, y)$ peut se traduire par l'expression de l'algèbre relationnelle suivante : $\mathbf{q} \setminus (\sqcap_{1/2}(\mathbf{p})) = \varnothing$. Pour modéliser les expressions de l'algèbre relationnelle et l'évaluation de ces dernières par un système de réécriture à la racine, nous introduisons les symboles suivants :

$$_ \triangleright _ : \mathsf{OperationsList} \times \mathsf{ArgumentsList} \rightarrow \mathsf{Computation}$$

symbolisant l'application de l'opération en-tête de la « pile d'opérations » en partie gauche du symbole $\triangleright$ aux arguments en tête de la « pile des arguments » en partie droite du symbole $\triangleright$. Les termes de sorte OperationsList sont des listes de termes de sorte Operation et les termes de sorte ArgumentsList sont des listes de termes de sorte Argument. Pour des raisons de lisibilité, nous utiliserons les mêmes symboles de constructeurs de listes (:: et []). Ainsi, Σ_{tco}^{AR} contient les symboles suivants de sorte Operation :

- **union** décrivant l'union de deux listes,
- **diff** décrivant la différences entre deux listes,
- $\mathbf{prod}_{k_1,k_2}$ décrivant le produit cartésien entre une liste de k_1-uplets et une liste de k_2-uplets,
- $\mathbf{proj}_{k_1,\ldots,k_m/k}$ décrivant la projection des k-uplets d'une liste sur les composantes $k_1, \ldots, k_m$,
- $\mathbf{join}_{k_1,k_2/k,k'}$ décrivant la jointure d'une liste de k-uplets et d'une liste de k'-uplets lorsque la $k_1{}^{\text{ème}}$ composante des tuples de la première liste coïncide avec la $k_2{}^{\text{ème}}$ composante des tuples de la seconde,
- **empty**, modélisant le test du caractère vide d'une liste,
- **isIn**, modélisant le test d'appartenance d'un élément à une liste,
- **first**, **min**, **max** réfèrant respectivement au premier, plus petit et plus grand élément d'une liste,
- $\mathbf{inf}_{\mathbf{Nat}}$, $\mathbf{inf}_{\mathbf{BoolList}}$, représentant respectivement les tests d'infériorité pour les ordres évoquées dans la section précédente,
- $\mathbf{eq}_{\mathsf{s}}$, décrivant le test d'égalité entre deux termes de sorte s,
- $\mathbf{select}_{k_1,k_2/k}$ décrivant la sélection (restriction) des k-uplets d'une liste à ceux dont la comparaison entre les $k_1{}^{\text{ème}}$ et $k_2{}^{\text{ème}}$ composantes correspond au comparateur passé en paramètre (faisant référence à **eq**, $\mathbf{inf}_{\mathbf{Nat}}$ ou $\mathbf{inf}_{\mathbf{BoolList}}$),
- $\mathbf{select}_{k'/k}$ décrivant la sélection (restriction) des k-uplets d'une liste à ceux dont la comparaison de la $k^{\text{ème}}$ composante avec le tuple en argument correspond au comparateur passé en paramètre,

- $\mathbf{swap_{k/k'}}$ décrit l'inversion des $k^{\text{ème}}$ et $k'^{\text{ème}}$ arguments dans la liste des arguments.

Le terme $\gamma_{\mathfrak{tco}}(\varphi)$ s'écrit alors : $\mathbf{proj_{1/2}} :: \mathbf{swap_{1/2}} :: \mathbf{diff} :: \mathbf{empty} :: [\,] \triangleright x_p :: x_q :: [\,]$. Pour permettre au lecteur de se faire une idée précise du système de réécriture $\mathcal{R}_{AR}$ sans toutefois en énumérer toutes les règles, les règles de réécriture associées aux principales opérations sont données par la figure 4.1. Elles ne présentent aucune difficulté particulière et sont construites à partir des algorithmes fonctionnels effectuant les opérations usuelles sur les listes. Pour obtenir un système de réécriture à la racine, les opérations à effectuer sont regroupées sous la forme d'une liste d'opérations et les opérandes sous regroupés sous la forme d'une liste d'arguments. Le système de réécriture obtenu fournit un processus d'évaluation de toute formule sûre sur un environnement donné à partir du terme $\sigma^{\eta}_{\varphi}(\gamma_{\mathfrak{tco}}(\varphi))$.

❞

Exemple 4.28. Pour bien comprendre le principe de fonctionnement du système $\mathcal{R}_{AR}$, étudions la réduction du terme $\sigma^{\eta}_{\varphi}(\gamma_{\mathfrak{tco}}(\varphi))$ où $\varphi = \forall x, \mathbf{q}(x) \Rightarrow \exists y, \mathbf{p}(x, y)$ lorsque $\mathbf{p}_\eta = \{(\mathbf{a}, \mathbf{b})\}$ et $\mathbf{q}_\eta = \{\mathbf{a}\}$.

$$\underline{\mathbf{proj_{1/2}}} :: \mathbf{swap_{1/2}} :: \mathbf{diff} :: \mathbf{empty} :: [\,] \triangleright \underline{\mathbf{cons}(\langle \mathbf{a}, \mathbf{b} \rangle_{\mathbf{2}}, \mathbf{nil})} :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{proj}^{\mathbf{aux}}_{\mathbf{1/2}}} :: \mathbf{swap_{1/2}} :: \mathbf{diff} :: \mathbf{empty} :: [\,] \triangleright \underline{\mathbf{cons}(\langle \mathbf{a}, \mathbf{b} \rangle_{\mathbf{2}}, \mathbf{nil}) :: \mathbf{nil}} :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{proj}^{\mathbf{aux}}_{\mathbf{1/2}}} :: \mathbf{swap_{1/2}} :: \mathbf{diff} :: \mathbf{empty} :: [\,] \triangleright \underline{\mathbf{nil}} :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{swap_{1/2}}} :: \mathbf{diff} :: \mathbf{empty} :: [\,] \triangleright \underline{\mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil})} :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{diff}} :: \mathbf{empty} :: [\,] \triangleright \underline{\mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil})} :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{diff}^{\mathbf{aux}}} :: \mathbf{empty} :: [\,] \triangleright \underline{\mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: \mathbf{nil}} :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{isIn}} :: \mathbf{diff}^{\mathbf{if}} :: \mathbf{empty} :: [\,] \triangleright \underline{\langle a \rangle_1 :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil})} :: \langle \mathbf{a} \rangle_{\mathbf{1}} :: \mathbf{nil} :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: \mathbf{nil} :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{eq}} :: \mathbf{or} :: \mathbf{isIn} :: \mathbf{diff}^{\mathbf{if}} :: \mathbf{empty} :: [\,] \triangleright \underline{\langle a \rangle_1 :: \langle a \rangle_1} :: \langle a \rangle_1 :: \mathbf{nil} :: \langle \mathbf{a} \rangle_{\mathbf{1}} :: \mathbf{nil} :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: \mathbf{nil} :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{eq}} :: \mathbf{or} :: \mathbf{isIn} :: \mathbf{diff}^{\mathbf{if}} :: \mathbf{empty} :: [\,] \triangleright \underline{a :: a} :: \langle a \rangle_1 :: \mathbf{nil} :: \langle \mathbf{a} \rangle_{\mathbf{1}} :: \mathbf{nil} :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: \mathbf{nil} :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{or} :: \mathbf{isIn}} :: \mathbf{diff}^{\mathbf{if}} :: \mathbf{empty} :: [\,] \triangleright \underline{\top :: \langle a \rangle_1 :: \mathbf{nil}} :: \langle \mathbf{a} \rangle_{\mathbf{1}} :: \mathbf{nil} :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: \mathbf{nil} :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{diff}^{\mathbf{if}}} :: \mathbf{empty} :: [\,] \triangleright \underline{\top :: \langle \mathbf{a} \rangle_{\mathbf{1}}} :: \mathbf{nil} :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil}) :: \mathbf{nil} :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{diff}^{\mathbf{aux}}} :: \mathbf{empty} :: [\,] \triangleright \underline{\mathbf{nil} :: \mathbf{cons}(\langle \mathbf{a} \rangle_{\mathbf{1}}, \mathbf{nil})} :: \mathbf{nil} :: [\,]$$
$$\downarrow$$
$$\underline{\mathbf{empty}} :: [\,] \triangleright \underline{\mathbf{nil}} :: [\,]$$
$$\downarrow$$
$$\underline{[\,]} \triangleright \underline{\top} :: [\,]$$
$$\downarrow$$
$$\mathbf{done}(\top)$$

(Les éléments soulignés correspondent aux symboles permettant de déterminer la règle de réécriture à appliquer).

$$[\,] \triangleright x :: q \;\rightarrow\; \mathbf{done}(x)$$

$$\begin{array}{ll}
\mathbf{and} :: q \triangleright \top :: q' & \rightarrow\; q \triangleright q' \\
\mathbf{and} :: h :: q \triangleright \bot :: x :: q' & \rightarrow\; q \triangleright \bot :: q' \\
\mathbf{or} :: q \triangleright \bot :: q' & \rightarrow\; q \triangleright q' \\
\mathbf{or} :: \mathbf{op} :: q \triangleright \top :: x_1 :: \ldots :: x_m :: q' & \rightarrow\; q \triangleright \top :: q' \quad \textit{pour tout opérateur } \mathbf{op} \textit{ d'arité } m
\end{array}$$

$$\begin{array}{ll}
\mathbf{eq} :: q \triangleright \mathbf{f}(x_1, \ldots, x_n) :: \mathbf{g}(y_1, \ldots, y_m) :: q' & \rightarrow\; q \triangleright \bot :: q' \\
\quad \textit{pour tous symboles } \mathbf{f} \neq \mathbf{g} & \\
\mathbf{eq} :: q \triangleright \mathbf{f}() :: \mathbf{f}() :: q' & \rightarrow\; q \triangleright \top :: q' \\
\quad \textit{pour tout symbole } \mathbf{f} \textit{ d'arité } 0 & \\
\mathbf{eq} :: q \triangleright \mathbf{f}(x_1, \ldots, x_n) :: \mathbf{f}(y_1, \ldots, y_n) :: q' & \\
\quad \rightarrow\; \mathbf{eq} :: \mathbf{and} :: \mathbf{eq} :: \ldots :: \mathbf{and} :: \mathbf{eq} :: q \triangleright x_1 :: y_1 :: \ldots :: x_n :: y_n :: q' & \\
\qquad \textit{pour tout symbole } \mathbf{f} \textit{ d'arité } n \neq 0 &
\end{array}$$

$$\begin{array}{ll}
\mathbf{proj}_{\mathbf{k_1},\ldots,\mathbf{k_m}/\mathbf{k}} :: q \triangleright l :: q' & \rightarrow\; \mathbf{proj}^{\mathbf{aux}}_{\mathbf{k_1},\ldots,\mathbf{k_m}/\mathbf{k}} :: q \triangleright l :: \mathbf{nil} :: q' \\
\mathbf{proj}^{\mathbf{aux}}_{\mathbf{k_1},\ldots,\mathbf{k_m}/\mathbf{k}} :: q \triangleright \mathbf{nil} :: q' & \rightarrow\; q \triangleright q' \\
\mathbf{proj}^{\mathbf{aux}}_{\mathbf{k_1},\ldots,\mathbf{k_m}/\mathbf{k}} :: q \triangleright \mathbf{cons}(\langle\, x_1, \ldots, x_k \,\rangle_k, z) :: u :: q' & \\
\quad \rightarrow\; \mathbf{proj}^{\mathbf{aux}}_{\mathbf{k_1},\ldots,\mathbf{k_m}/\mathbf{k}} :: q \triangleright z :: \mathbf{cons}(\langle\, x_{k_1}, \ldots, x_{k_m} \,\rangle_m, u) :: q' &
\end{array}$$

$$\begin{array}{ll}
\mathbf{isIn} :: q \triangleright x :: \mathbf{nil} :: q' & \rightarrow\; q \triangleright \bot :: q' \\
\mathbf{isIn} :: q \triangleright x :: \mathbf{cons}(y, z) :: q' & \rightarrow\; \mathbf{eq_s} :: \mathbf{or} :: \mathbf{isIn} :: q \triangleright x :: y :: x :: z :: q'
\end{array}$$

$$\begin{array}{ll}
\mathbf{diff} :: q \triangleright l :: l' :: q' & \rightarrow\; \mathbf{diff}^{\mathbf{aux}} :: q \triangleright l :: l' :: \mathbf{nil} :: q' \\
\mathbf{diff}^{\mathbf{aux}} :: q \triangleright \mathbf{nil} :: l :: q' & \rightarrow\; q \triangleright q' \\
\mathbf{diff}^{\mathbf{aux}} :: q \triangleright \mathbf{cons}(x, y) :: z :: u :: q' & \\
\quad \rightarrow\; \mathbf{isIn} :: \mathbf{diff}^{\mathbf{if}} :: q \triangleright x :: z :: x :: y :: z :: u :: q' & \\
\mathbf{diff}^{\mathbf{if}} :: q \triangleright \bot :: x :: y :: z :: u :: q' & \rightarrow\; \mathbf{diff}^{\mathbf{aux}} :: q \triangleright y :: z :: \mathbf{cons}(x, u) :: q' \\
\mathbf{diff}^{\mathbf{if}} :: q \triangleright \top :: x :: y :: z :: u :: q' & \rightarrow\; \mathbf{diff}^{\mathbf{aux}} :: q \triangleright y :: z :: u :: q'
\end{array}$$

$$\begin{array}{ll}
\mathbf{prod}_{\mathbf{k},\mathbf{k'}} :: q \triangleright l :: l' :: q' & \rightarrow\; \mathbf{prod}^{\mathbf{aux}}_{\mathbf{k},\mathbf{k'}} :: q \triangleright l :: l' :: \mathbf{nil} :: q' \\
\mathbf{prod}^{\mathbf{aux}}_{\mathbf{k},\mathbf{k'}} :: q \triangleright \mathbf{nil} :: l :: q' & \rightarrow\; q \triangleright q' \\
\mathbf{prod}^{\mathbf{aux}}_{\mathbf{k},\mathbf{k'}} :: q \triangleright \mathbf{cons}(x, y) :: z :: u :: q' & \\
\quad \rightarrow\; \mathbf{dist}^{\mathbf{aux}}_{\mathbf{k},\mathbf{k'}} :: \mathbf{prod}^{\mathbf{aux}}_{\mathbf{k},\mathbf{k'}} :: q \triangleright x :: z :: y :: z :: u :: q' & \\
\mathbf{dist}^{\mathbf{aux}}_{\mathbf{k},\mathbf{k'}} :: q \triangleright x :: \mathbf{nil} :: y :: z :: u :: q' & \rightarrow\; q \triangleright y :: z :: u :: q' \\
\mathbf{dist}^{\mathbf{aux}}_{\mathbf{k},\mathbf{k'}} :: q \triangleright \langle\, x_1, \ldots, x_k \,\rangle_k :: \mathbf{cons}(\langle\, v_1, \ldots, v_{k'} \,\rangle_{k'}, v) :: y :: z :: u :: q' & \\
\quad \rightarrow\; \mathbf{dist}^{\mathbf{aux}}_{\mathbf{k},\mathbf{k'}} :: q \triangleright \langle\, x_1, \ldots, x_k \,\rangle_k :: v :: y :: z :: \mathbf{cons}(\langle\, x_1, \ldots, x_k, v_1, \ldots, v_{k'} \,\rangle_{k+k'}, u) :: q' &
\end{array}$$

$$\begin{array}{ll}
\mathbf{first} :: q \triangleright \mathbf{cons}(x, l) :: q' & \rightarrow\; q \triangleright x :: q' \\
\mathbf{empty} :: q \triangleright \mathbf{cons}(x, l) :: q' & \rightarrow\; q \triangleright \bot :: q' \\
\mathbf{empty} :: q \triangleright \mathbf{nil} :: q' & \rightarrow\; q \triangleright \top :: q'
\end{array}$$

FIGURE 4.1 – Règles associées aux principales opérations de l'algèbre relationnelle

À noter que l'intérêt du système de réécriture obtenu n'est pas l'efficacité de l'évaluation mais sa spécification particulière permettant de tirer profit des techniques de vérification précédemment évoquées ainsi que des techniques de preuves inductives (ce point sera discuté dans la section 4.3).

4.2.3 Systèmes de sécurité

Nous pouvons désormais montrer que nous pouvons construire, grâce à la proposition 4.21, un système de réécriture à la racine, basé sur les constructeurs (c.-à-d. tel que les termes composés uniquement de constructeurs sont irréductibles), simulant la relation de transition d'un système de sécurité donné.

Proposition 4.29. Soit $\mathfrak{teo}$ une théorie de sécurité et $\mathfrak{S}$ un système de sécurité sur $\mathfrak{teo}$. Il existe un système de réécriture $\mathcal{R}_{\mathfrak{S}}$ à la racine convergent tel que pour tout environnement η et toute séquence d'événements $evts = \langle\,(act_1, d_1), \ldots, (act_n, d_n)\,\rangle$, $\eta \xrightarrow{evts}_{\mathfrak{S}} \eta'$ si et seulement si

$$\gamma_{\mathfrak{teo}}(\eta) \otimes (act_1, d_1) :: \ldots :: (act_n, d_n) :: [\,] \xrightarrow{*}_{\mathcal{R}_{\mathfrak{S}}} \gamma_{\mathfrak{teo}}(\eta') \otimes [\,]$$

(où $\otimes$ est un nouveau symbole binaire modélisant l'application d'une séquence d'événements sur un environnement).

Pour démontrer cette propriété, nous avons recours aux deux lemmes suivants :

Lemme 4.30. Soient $(E, \leq_E)$ un ensemble totalement ordonné et Φ un isomorphisme d'ordre de $(E, \leq_E)$ dans $(F \subseteq \mathbb{N}, \leq_{\mathbb{N}})$. Pour tout élément $e \notin E$ associé à la contrainte $e_{before} \leq e \leq e_{after}$, (pour e_{before} et e_{after} dans E) l'application Φ' définie par :

$$\begin{cases} \Phi'(e) = \Phi(e_{after}) \\ \forall e' \in E, e' \leq e_{before}, \Phi'(e') = \Phi(e') \\ \forall e' \in E, e_{after} \leq e', \Phi'(e') = \Phi(e') + 1 \end{cases}$$

est un isomorphisme de la mise à jour de $(E, \leq_E)$ pour la contrainte $e_{before} \leq e \leq e_{after}$ dans $(F' \subseteq \mathbb{N}, \leq_{\mathbb{N}})$.

Lemme 4.31. Soient $(E, \leq_E)$ un ensemble partiellement ordonné et Φ un isomorphisme d'ordre de $(E, \leq_E)$ dans $(F \subseteq \mathbb{B}^n, \leq_\times)$. Pour tout élément $e \notin E$ associé à la contrainte $e^1_{before}, \ldots, e^m_{before} \leq e \leq e^1_{after}, \ldots, e^k_{after}$, (on suppose que les e^i_{before} sont sont deux à deux incomparables pour $\leq_E$, de même que les e^j_{after} et que $e^i_{before} <_E e^j_{after}$ quels que soient i et j) l'application Φ' définie par :

$$\begin{cases} \Phi'(e) = (\Phi(e_{max}), 0)\ \textit{si}\ m > 1\ \textit{et}\ \Phi'(e) = (\Phi(e_{max}), 1)\ \textit{sinon} \\ \forall e' \in E, e' \leq_E e_{max}, \Phi'(e') = (\Phi(e'), 0) \\ \forall e' \in E, e_{min} \leq_E e', \Phi'(e') = (\Phi(e'), 1) \\ \forall e' \in E, e_{min} \bowtie_E e', \Phi'(e') = (\Phi(e'), 0) \end{cases}$$

avec $\Phi(e_{max}) = (\max_{i=1\ldots m}(b_1^i), \ldots, \max_{i=1\ldots m}(b_n^i))$ où $(b_1^i, \ldots, b_n^i) = \Phi(e^i_{before})$ et $\Phi(e_{min}) = (\max_{j=1\ldots k}(b_1^j), \ldots, \max_{i=j\ldots k}(b_n^j))$ où $(b_1^j, \ldots, b_n^j) = \Phi(e^j_{after})$, est un isomorphisme de la mise à jour de $(E, \leq_E)$ pour la contrainte $e^1_{before}, \ldots, e^m_{before} \leq e \leq e^1_{after}, \ldots, e^k_{after}$, dans $(F' \subseteq \mathbb{B}^{n+1}, \leq_\times)$.

❝ *Démonstration (Lemmes).* Le lemme 4.30 est trivial. Le lemme 4.31 est quant à lui basé sur les constats suivants :

- si $m > 1$, alors $\Phi(e_{max}) >_\times \Phi(e^i_{before})$ pour tout i,
- si $k > 1$, alors $\Phi(e_{min}) <_\times \Phi(e^j_{after})$ pour tout j,
- pour tous $\alpha, \beta \in \mathbb{B}^n$, les trois inégalités suivantes sont équivalentes :

$$\begin{cases} \alpha <_\times \beta \\ (\alpha, 0) <_\times (\alpha, 1) <_\times (\beta, 1) \\ (\alpha, 0) <_\times (\beta, 0) <_\times (\beta, 1) \end{cases}$$

❞

❝ *Démonstration (Proposition 4.29).* Dans la suite, étant donnés une signature $\Sigma = (\mathcal{S}, \mathcal{F})$, deux $\mathcal{S}$-ensembles de variables disjoints $\mathcal{X}_1$ et $\mathcal{X}_2$, un système de réécriture $\mathcal{R}$ sur Σ (et $\mathcal{X}_1$) ainsi qu'un contexte C sur $\Sigma' = (\mathcal{S}, \mathcal{F} \cup \{\mathcal{X}_2\})$ (où les variables de $\mathcal{X}_2$ sont vus comme des constantes), on note $C[\mathcal{R}]$ le système de réécriture contenant les règles $C[lhs] \to C[rhs]$ pour toute règle $lhs \to rhs$ de $\mathcal{R}$. On modifie la signature $\Sigma^{AR}_{\mathfrak{tco}}$ en considérant les symboles suivants :

$$\begin{array}{rcll} _ \otimes _ & : & \mathsf{Environment} \times \mathsf{Events} & \to \mathsf{State} \\ \mathbf{check_r} & : & \mathsf{Environment} \times \mathsf{Computation} \times \mathsf{Events} & \to \mathsf{State} \\ \mathbf{map_r} & : & \mathsf{Environment} \times \mathsf{TupleList} \times \mathsf{Facts} \times \mathsf{Events} & \to \mathsf{State} \\ \mathbf{replace_p} & : & \mathsf{Facts_p} \times \mathsf{Computation} \times \mathsf{Events} & \to \mathsf{State} \end{array}$$

pour tout prédicat $\mathbf{p}$ de $\tilde{\Sigma}^{Env}_{\mathfrak{tco}}$ et toute règle de transition r de $\mathfrak{S}$. Le système $\mathcal{R}_{\mathfrak{S}}$ contient, pour toute règle de transition r,

- $C_r[\mathcal{R}_{AR}]$ où $C_r = \mathbf{check_r}(x, \Box)$,
- si $r = (act, d) : \underbrace{\mathbf{p}(t_1, \ldots, t_n)}_{C} \leftarrow \alpha$:

$$\begin{array}{rcl} x \otimes (act, d) :: q_{events} & \to & \mathbf{check_r}(x, \gamma_{\mathfrak{tco}}(\alpha), q_{events}) \\ \mathbf{check_r}(x, [\,] \triangleright r :: q, q_{events}) & \to & \mathbf{map_r}(x, r, [\,], q_{events}) \\ \mathbf{map_r}(x, \langle x_1, \ldots, x_n \rangle :: q, l, q_{events}) & \to & \mathbf{map_r}(x, q, C[x_1, \ldots, x_n] :: l, q_{events}) \\ \mathbf{map_r}(\langle x_{p_1}, \ldots, x_{p_n} \rangle, [\,], l) & & \\ \to \mathbf{replace_p}(\langle x_{p_1}, \ldots, x_{p_n} \rangle, \mathbf{union} :: [\,] \triangleright x_p :: l, q_{events}) & & \end{array}$$

Un terme de la forme $x \otimes l_{events}$ représente le calcul de l'état résultant de l'application de la séquence d'événements décrite par la liste l_{events}. Pour effectuer ce calcul, la première étape consiste à déterminer les solutions de α dans l'environnement courant. C'est l'objet de la réduction des termes de la forme $\mathbf{check_r}(x, \gamma_{\mathfrak{tco}}(\alpha), _)$ vers un terme de la forme $\mathbf{check_r}(_, [\,] \triangleright r :: _, _)$ dont le sous-terme caractérisé par r contient la liste des solutions. À partir de ces

solutions, on peut construire l'ensemble des atomes clos à ajouter à l'environnement, c'est à dire l'ensemble des $\sigma_i(\mathbf{p}(t_1, \ldots, t_n))$ où à chaque élément de la liste des solutions correspond un σ_i (les atomes $\sigma_i(\mathbf{p}(t_1, \ldots, t_n))$ sont construits par la règle $\mathbf{map_r}(_, \langle\, x_1, \ldots, x_n \,\rangle :: _, _, _) \to \mathbf{map_r}(_, _, C[x_1, \ldots, x_n] :: _, _))$.

- si $r = (act, d) : \neg \underbrace{\mathbf{p}(t_1, \ldots, t_n)}_{C} \leftarrow \alpha$:

$$\begin{aligned}
x \otimes (act, d) :: q_{events} &\to \mathbf{check_r}(x, \gamma_{\mathtt{tco}}(\alpha), q_{events}) \\
\mathbf{check_r}(x, [\,] \ \triangleright\ r :: q, q_{events}) &\to \mathbf{map_r}(x, r, [\,], q_{events}) \\
\mathbf{map_r}(x, \langle\, x_1, \ldots, x_n \,\rangle :: q, l, q_{events}) &\to \mathbf{map_r}(x, q, C[x_1, \ldots, x_n] :: l, q_{events}) \\
\mathbf{map_r}(\langle\, x_{p_1}, \ldots, x_{p_n} \,\rangle, [\,], l, q_{events}) & \\
&\to \mathbf{replace_p}(\langle\, x_{p_1}, \ldots, x_{p_n} \,\rangle, \mathbf{diff} :: [\,] \ \triangleright\ x_p :: l, q_{events})
\end{aligned}$$

- si $r = (act, d) : \mathbf{p}(t_1, \ldots, t_n) \leftarrow \alpha \mid \begin{cases} \mathbf{inf}(e^1_{before}, e) \\ \vdots \\ \mathbf{inf}(e^m_{before}, e) \end{cases} \wedge \begin{cases} \mathbf{inf}(e, e^1_{after}) \\ \vdots \\ \mathbf{inf}(e, e^k_{after}) \end{cases}$,

  les règles devenant plus techniques et plus nombreuses, nous nous contenterons d'expliquer le principe de ces dernières. Les règles évoquées pour le cas $r = (act, d) : \mathbf{p}(t_1, \ldots, t_n) \leftarrow \alpha$ sont adaptées pour permettre une étape supplémentaire entre la vérification de la contrainte (réduction des termes dont la racine est $\mathbf{check_r}$) et le calcul des atomes à ajouter (réduction des termes dont la racine est $\mathbf{map_r}$). Chaque tuple solution de la contrainte (c'est-à-dire chaque élément de la liste r dans $\mathbf{check_r}(x, [\,] \ \triangleright\ r :: q, q_{events})$), contient les valeurs des $\Phi(e^i_{before})$ et des $\Phi(e^j_{after})$ (à des positions déterminables statiquement). À partir de ces informations, une étape de mise à jour de tous les termes représentant un élément appartenant à l'ensemble ordonné considéré, suivant les lemmes 4.30 et 4.31, est effectuée. Cette étape ajoute également une composante aux tuples résultats de la contrainte qui ne contenaient jusqu'alors pas les valeurs des variables libres de β (c'est-à-dire les nouveaux éléments à « insérer » dans l'ordre). À l'issue de cette étape, les tuples solutions sont « complets » (ils contiennent les valeurs de toutes les variables libres) et le calcul des atomes clos à ajouter peut être entamé (réduction vers un terme de racine $\mathbf{map_r}$).

ainsi que, pour tout symbole de prédicat $\mathbf{p}$ de $\tilde{\Sigma}^{Env}_{\mathtt{tco}}$ tel que $\mathcal{C}_{\mathtt{tco}}(\mathbf{p}) = empty$, $C_p[\mathcal{R}_{AR}]$ où $C_p = \mathbf{replace_p}(x, \Box)$ et :

$$\begin{aligned}
&\mathbf{replace_p}(x, \mathbf{done}(\langle\, x_{p_1}, \ldots, x_{p_n} \,\rangle), q_{events}) \\
&\qquad \to \left\langle\, x_{p_1}, \ldots, x_{p_{i-1}}, x, x_{p_{i+1}}, \ldots, x_{p_n} \,\right\rangle \otimes q_{events}
\end{aligned}$$

(où i est l'indice correspondant à $\mathbf{p}$). ❞

4.2.4 Politiques de sécurité

De la même façon que nous avons proposé une sémantique de la relation de transition d'un système de sécurité sous la forme d'un système de réécriture à la racine, nous nous attachons désormais à présenter un tel système pour décrire une politique de sécurité décrite dans notre formalisme.

Proposition 4.32. Soit $\mathfrak{teo}$ une théorie de sécurité et $\mathfrak{pol}$ une politique de sécurité sur $\mathfrak{teo}$. Il existe un système de réécriture $\mathcal{R}_{\mathfrak{pol}}$ à la racine tel que :

- $\gamma_{\mathfrak{teo}}(\eta) \vdash \mathit{action} \xrightarrow{*}_{\mathcal{R}_{\mathfrak{pol}}} \mathit{decision}$ ssi $\mathit{action} \xrightarrow{*}{}^{\mathfrak{pol}}_{\eta} \mathit{decision}$,
- $\rightarrow_{\mathcal{R}_{\mathfrak{pol}}}$ termine (resp. est confluent sur les termes clos, resp. suffisamment complet) ssi $\mathfrak{pol}$ termine (resp. est cohérente, resp. complète).

❝ *Démonstration.* On ajoute à la signature $\Sigma^{AR}_{\mathfrak{teo}}$ les symboles suivants :

$$
\begin{array}{rcll}
_ \vdash _ & : & \mathsf{Environment} \times \mathsf{Action} & \rightarrow \mathsf{Request} \\
_ \vdash^{aux} _ \, , \, _ & : & \mathsf{Environment} \times \mathsf{Action} \times \mathsf{Trace} & \rightarrow \mathsf{Request} \\
\mathbf{check_r} & : & \mathsf{Environment} \times \mathsf{Computation} \times \mathsf{Trace} & \rightarrow \mathsf{Request} \\
\mathbf{undefined} & : & & \rightarrow \mathsf{Request} \\
\langle _, \ldots, _ \rangle & : & \mathsf{Bool} \times \ldots \times \mathsf{Bool} & \rightarrow \mathsf{Trace}
\end{array}
$$

pour toute règle r de $\mathfrak{pol}$. $\mathcal{R}_{\mathfrak{pol}}$ contient le système $C[\mathcal{R}_{AR}]$ avec $C = \mathbf{check_r}(\Box, x, y, z)$ ainsi que les règles suivantes :

$$
\begin{array}{rcl}
x \vdash y & \rightarrow & x \vdash^{aux} y \, , \, \langle \bot, \ldots, \bot \rangle \\
x \vdash^{aux} y \, , \, \langle t_1, \ldots, t_m \rangle & \rightarrow & \mathbf{check_r}(x, \gamma_{\mathfrak{teo}}(y = \mathit{action} \wedge \varphi), \langle t'_1, \ldots, t'_m \rangle) \\
\mathbf{check_r}(x, [\,] \triangleright \langle y_1, \ldots, y_n \rangle :: [\,], z) & \rightarrow & \begin{cases} x \vdash \mathit{rhs}[y_1, \ldots, y_n] \ \mathit{si}\ \mathit{rhs} \notin \mathcal{D}_{\mathfrak{teo}} \\ \mathit{rhs} \ \mathit{si}\ \mathit{rhs} \in \mathcal{D}_{\mathfrak{teo}} \end{cases} \\
\mathbf{check_r}(x, [\,] \triangleright [\,], z) & \rightarrow & x \vdash^{aux} y \, , \, z \\
x \vdash^{aux} y \, , \, \langle \top, \ldots, \top \rangle & \rightarrow & \mathbf{undefined}
\end{array}
$$

pour toute règle $r = \mathit{action}[y_1, \ldots, y_n] \xrightarrow{\varphi} \mathit{rhs}[y_1, \ldots, y_n]$ et où $\langle t_1, \ldots, t_m \rangle$ est le terme de sorte Trace tel que t_i est égal à $\bot$ si i est l'indice de la règle r, $\top$ si i est l'indice d'une règle $r' <_{\mathfrak{pol}} r$, et une variable sinon tandis que t'_i est égal à t_i sauf si i est l'indice de r auquel cas $t'_i = \top$. Intuitivement, le terme $\langle t_1, \ldots, t_m \rangle$ signifie que la règle correspondant à l'indice i a été « testée » (et n'est pas applicable) ssi $t_i = \top$. La présence d'un terme de sorte Trace est nécessaire pour que le système de réécriture soit confluent si la spécification initiale de la politique est cohérente. À noter que les variables libres de la formule $y = \mathit{action}[y_1, \ldots, y_n] \wedge \varphi$ dans la seconde règle (y est instancié) correspondent aux variables du motif *action*, c'est-à-dire $\{y_1, \ldots, y_n\}$. En conséquence, le résultat de l'évaluation de cette formule est soit la liste vide (si l'instance de y ne filtre pas le motif *action* ou si φ n'est pas vérifiée) soit le tuple $\langle t_1, \ldots, t_n \rangle$ tel que $\mathit{action}[t_1, \ldots, t_n]$ soit égal à l'instanciation de y. L'équivalence de la terminaison des systèmes de réécriture $\rightarrow_{\mathcal{R}_{\mathfrak{pol}}}$ et $\rightarrow^{\mathfrak{pol}}_{\eta}$ est évidente dans la mesure où le système $\mathcal{R}_{AR}$ termine et que les systèmes sont des systèmes de réécriture à la racine. La non-terminaison de $\rightarrow_{\mathcal{R}_{\mathfrak{pol}}}$ ne peut provenir que d'une boucle provoquée par la réécriture d'une action en une autre action, de même pour $\rightarrow^{\mathfrak{pol}}_{\eta}$. ❞

4.2.5 Système sécurisé

Le système de réécriture simulant la relation de transition d'un système sécurisé s'obtient aisément à partir de la sémantique du système de sécurité et de la politique de sécurité à partir desquels il est construit.

Proposition 4.33. Soient $\mathfrak{teo}$ une théorie de sécurité, $\mathfrak{S}$ un système de sécurité sur $\mathfrak{teo}$ et $\mathfrak{pol}$ une politique de sécurité. Il existe un système de réécriture $\mathcal{R}_{\mathfrak{S},\mathfrak{pol}}$ à la racine convergent tel que pour tout environnement η et toute séquence d'actions $\langle act_1, \ldots, act_n \rangle$, $\eta \xrightarrow{act_1}_{\mathfrak{S}_{\mathfrak{pol}}} \ldots \xrightarrow{act_n}_{\mathfrak{S}_{\mathfrak{pol}}} \eta'$ si et seulement si $\gamma_{\mathfrak{teo}}(\eta) \mathbin{\tilde{\otimes}} act \xrightarrow{*}_{\mathcal{R}_{\mathfrak{S},\mathfrak{pol}}} \gamma_{\mathfrak{teo}}(\eta') \otimes [\,]$ (où $\tilde{\otimes}$ est un nouveau symbole caractérisant l'application d'une séquence d'actions à partir d'un environnement donné).

❝ *Démonstration.* Pour simplifier, on ne considère dans la preuve qu'une seule action (au lieu d'une séquence d'actions). $\mathcal{R}_{\mathfrak{S},\mathfrak{pol}}$ contient alors la règle

$$x \mathbin{\tilde{\otimes}} act \quad \to \quad x \otimes (act, x \vdash act) :: [\,]$$

ainsi que $C[\mathcal{R}_{\mathfrak{pol}}]$ avec $C = x \otimes (y, \Box) :: q$ et $\mathcal{R}_{\mathfrak{S}}$. Pour obtenir le système simulant la relation de transition pour des séquences d'actions, il suffit d'adapter les règles comme dans $\mathcal{R}_{\mathfrak{S}}$ pour conserver la « queue » de la séquence d'actions. ❞

4.3 Vérification

Le but de donner une sémantique basée sur la réécriture des spécifications de systèmes et de politiques est de permettre l'analyse automatique de ces spécifications par des outils éprouvés. Dans cette section, nous allons montrer comment les propriétés des systèmes et des politiques de sécurité s'expriment au moyen de propriétés sur les systèmes de réécriture correspondants.

Les systèmes de réécriture obtenus possèdent deux propriétés particulières : d'une part, ils sont basés sur la notion de constructeurs et offrent une définition inductive des objets qu'ils spécifient, permettant de tirer profit des techniques de preuve par induction implicite [Huet and Hullot, 1982, Bouhoula, 1996, Falke and Kapur, 2006], [Bouhoula, 1998] et des outils correspondants [Bouhoula and Rusinowitch, 1995], [Stratulat, 2011]. D'autre part, les systèmes sont des systèmes de réécriture à la racine et permettent donc d'appliquer les techniques récemment développées dans [Rusu, 2010, Rocha and Meseguer, 2011, Rocha, 2011].

4.3.1 Propriétés de politiques

Nous avons évoqué dans la précédente section les propriétés fondamentales des politiques de sécurité. Il s'agit des propriétés de terminaison, de cohérence et de complétude. La proposition suivante rappelle une propriété déjà évoquée mais qu'il est pertinent de mentionner ici :

Proposition 4.34. Soit $\mathfrak{teo}$ une théorie de sécurité et $\mathfrak{pol}$ une politique de sécurité sur $\mathfrak{teo}$. $\mathfrak{pol}$ termine (resp. est cohérente, resp. complète) si et seulement si $\to_{\mathcal{R}_{\mathfrak{pol}}}$ termine (resp. est confluent sur les termes clos, resp. suffisamment complet).

La complétude suffisante et la confluence sur les termes clos peuvent être vérifiées à l'aide des techniques évoquées dans [Bouhoula, 2009] et implantées dans l'outil SPIKE [Bouhoula and Rusinowitch, 1995]. La méthode présentée dans ces travaux s'applique aux spécifications présentées sous la forme de systèmes de réécriture basés sur les constructeurs donc, en particulier, au système $\mathcal{R}_{\mathfrak{pol}}$. Un des intérêts de notre modélisation est d'offrir une définition inductive des modèles finis d'une formule donnée (via le système de réécriture $\mathcal{R}_{AR}$), permettant de tirer profit des techniques de preuve par induction implicite. Sans cette traduction, il aurait été nécessaire de combiner une méthode pour déterminer l'existence d'un modèle fini satisfaisant la conjonction de deux formules (pour déterminer les règles qui peuvent s'appliquer simultanément) avec une méthode montrant que l'application des deux règles concernées pour le modèle trouvé engendre le même résultat (confluence « sémantique »). Dans ce contexte, notre cadre nous semble être particulièrement attractif. La terminaison peut quant à elle être automatiquement vérifiée par une très large panoplie d'outils : CiME [Contejean et al., 2010], AProVE [Giesl et al., 2006], TTT [Hirokawa and Middeldorp, 2005] ou encore MTT [Durán et al., 2008].

4.3.2 Le problème de l'administration

Une question importante lors de l'analyse d'une politique de sécurité est ce que nous appelerons le « problème de l'administration ». Il consiste à déterminer si un ensemble d'utilisateurs en administre un autre (éventuellement identifique). Par « administrer » on entend la faculté de modifier les droits. Formellement, le problème peut être exprimé de la façon suivante :

Définition 4.35. Soit $\mathfrak{teo}$ une théorie de sécurité, $\mathfrak{S}$ un système de sécurité et $\mathfrak{pol}$ une politique de sécurité sur $\mathfrak{teo}$. On suppose que $\mathfrak{teo}$ contient une sorte s apparaissant dans le profil de toutes les actions (notons ω la position du sous-terme de sorte s dans chaque action). On dit que l'ensemble $A \subseteq \mathcal{T}(\Sigma_{\mathfrak{teo}})_{\mathsf{s}}$ **administre** l'ensemble $U \subseteq \mathcal{T}(\Sigma_{\mathfrak{teo}})_{\mathsf{s}}$ ssi il existe une action act telle que $act_{|\omega} \in U$, un environnement η et une séquence d'actions $adm_1, \ldots, adm_n$ telle que pour tout i, $adm_{i|\omega} \in A$ et

$$act \xrightarrow{*}{}^{\eta}_{\mathfrak{pol}} decision \neq decision' \; {}^{\eta'}_{\mathfrak{pol}}\xleftarrow{*} act$$

où $\eta \xrightarrow{adm_1,\ldots,adm_n}_{\mathfrak{S}_{\mathfrak{pol}}} \eta'$. Le problème de l'administration dans $\mathfrak{S}$ et $\mathfrak{pol}$ de U par A consiste à déterminer si A administre U (relativement à $\mathfrak{S}$ et $\mathfrak{pol}$).

C'est un problème souvent étudié dans le cadre de l'analyse des politiques basées sur les rôles [Sasturkar et al., 2006]. Un sous-problème connu du problème de l'administration est celui de la « sûreté » dans le modèle HRU.

Notre cadre est particulièrement adapté à l'étude de ce problème puisque ce dernier peut s'exprimer sur un système de réécriture à la racine obtenu à partir des systèmes présentés dans la section précédente sous la forme d'un problème d'accessibilité dont la résolution est traité dans [Rusu, 2010, Rocha and Meseguer, 2011, Rocha, 2011] pour la classe de systèmes de réécriture qui nous concerne. La définition d'un problème d'accessibilité est donné par la définition suivante :

Définition 4.36. Soit Σ une signature, $\mathcal{R}$ un système de réécriture sur Σ, t_0 un terme de $\mathcal{T}(\Sigma, \mathcal{X})$ et t un terme de $\mathcal{T}(\Sigma)$, on appelle **problème d'accessibilité** dans $\mathcal{R}$ relativement à t_0 et t le problème de l'existence d'une substitution close σ telle que $\sigma(t_0) \xrightarrow{*}_{\mathcal{R}} t$.

Pour cela, pour tout système de sécurité $\mathfrak{S}$, toute politique $\mathfrak{pol}$ sur une théorie de sécurité $\mathfrak{teo}$, et tous motifs t_A et t_U de la sorte considérée par le problème, on établit le système de réécriture $\mathcal{R}^{adm}_{\mathfrak{S},\mathfrak{pol}}$ constitué des règles :

$$\begin{array}{rcl} u \vdash_{adm} (x, y, z) & \to & (u \mathbin{\tilde{\otimes}} y) \vdash x \ ; \ u \vdash z \\ d \ ; \ d' & \to & \textbf{changed} \end{array}$$

pour tout couple de décisions $d \neq d'$, ainsi que $\begin{cases} C[\mathcal{R}_{\mathfrak{pol}}] \ \textit{avec} \ C = x \ ; \ \square \\ C'[\mathcal{R}_{\mathfrak{S}_{\tilde{\mathfrak{pol}}}}] \ \textit{avec} \ C' = \square \vdash x \ ; \ y \\ C''[\mathcal{R}_{\mathfrak{pol}}] \ \textit{avec} \ C'' = \square \ ; \ z \end{cases}$ où $\mathcal{R}_{\mathfrak{S}_{\tilde{\mathfrak{pol}}}}$ correspond au système de réécriture $\mathcal{R}_{\mathfrak{S}_{\mathfrak{pol}}}$ dans lequel les actions dont le sous-terme de position ω ne filtre pas t_A sont supprimées. Le problème de l'administration de U par A s'exprime par l'union des problèmes d'accessibilité suivants : existe-t-il une substitution close σ telle que

$$\sigma(u \vdash_{adm} (\mathbf{act}(x_1, \ldots, t_U, \ldots, x_n), y, \mathbf{act}(x_1, \ldots, t_U, \ldots, x_n))) \xrightarrow{*}_{\mathcal{R}^{adm}_{\mathfrak{S},\mathfrak{pol}}} \mathbf{changed}$$

pour tout symbole d'action **act**. Le système $\mathcal{R}^{adm}_{\mathfrak{S},\mathfrak{pol}}$ étant un système de réécriture à la racine, la résolution de ce problème peut être traité par les méthodes décrites dans [Rusu, 2010, Rocha and Meseguer, 2011, Rocha, 2011].

4.3.3 Comparaison de politique

La comparaison de politiques a déjà été évoquée dans le chapitre précédent. Rappelons la définition de l'équivalence entre deux politiques dans notre cadre.

Définition 4.37. Soient $\mathfrak{pol}_1$ et $\mathfrak{pol}_2$ deux politiques terminantes, complètes et cohérentes sur une théorie de sécurité $\mathfrak{teo}$. On dit que $\mathfrak{pol}_1$ et $\mathfrak{pol}_2$ sont **équivalentes** et l'on note $\mathfrak{pol}_1 \approx_{\mathfrak{teo}} \mathfrak{pol}_2$ si et seulement si pour tout environnement η sur $\mathfrak{teo}$, pour toute action act et pour toute décision d, $act \xrightarrow{*}{}^{\eta}_{\mathfrak{pol}_1} d$ ssi $act \xrightarrow{*}{}^{\eta}_{\mathfrak{pol}_2} d$.

Notons $\mathcal{R}_{\mathfrak{pol}_1,\mathfrak{pol}_2}$ le système de réécriture obtenu à partir de $\mathcal{R}_{\mathfrak{pol}_1}$ et $\mathcal{R}_{\mathfrak{pol}_2}$ en remplaçant la règle $x \ \vdash y \to x \ \vdash^{aux} y \ , \ \langle \bot, \ldots, \bot \rangle$ par les règles $x \ \vdash_1 y \to x \ \vdash^{aux} y \ , \ \langle \bot, \ldots, \bot \rangle$ et $x \ \vdash_2 y \to x \ \vdash^{aux} y \ , \ \langle \bot, \ldots, \bot \rangle$ respectivement. On a la proposition suivante :

Proposition 4.38. Soient $\mathfrak{pol}_1$ et $\mathfrak{pol}_2$ deux politiques terminantes, complètes et cohérentes sur une théorie de sécurité $\mathfrak{teo}$. $\mathfrak{pol}_1 \approx_{\mathfrak{teo}} \mathfrak{pol}_2$ si et seulement si $x \ \vdash_1 y = x \ \vdash_2 y$ est un théorème inductif de $\mathcal{R}_{\mathfrak{pol}_1,\mathfrak{pol}_2}$.

Cette proposition nous indique que l'on peut vérifier l'équivalence entre deux politiques de sécurité dynamiques en utilisant les techniques de preuves par induction implicite [Stratulat, 2005] implantée dans l'outil SPIKE-Prover [Stratulat, 2011]. Ceci est rendu possible par la forme du système de réécriture $\mathcal{R}_{\mathfrak{pol}_1,\mathfrak{pol}_2}$ (basé sur les constructeurs, terminant, suffisamment complet et confluent sur les termes clos).

4.3.4 Analyse par requêtage

Comme nous l'avons évoqué dans le chapitre précédent, il est souvent utile de pouvoir « interroger » une politique de sécurité en formulant des requêtes spécifiant partiellement les éléments à partir desquels la politique est définie. Ce type d'analyse a été présenté sous le nom d'analyse par requêtage ou encore d'analyse administrative. Dans le cas des politiques dynamiques, ce type d'analyse diffère légèrement de la présentation qui en a été faite dans le chapitre 3. En effet, dans le cas présent l'analyse peut correspondre à l'évaluation d'une requête contenant des inconnues dans un environnement complètement spécifié ou bien à l'évaluation d'une requête sous un certain nombre de contraintes spécifiant partiellement un environnement. Pour permettre ce type d'analyse dans notre cadre, il est nécessaire que les contraintes possèdent une forme particulière. Nous baptisons de $\mathfrak{teo}$-définissables les contraintes éligibles relativement à la théorie $\mathfrak{teo}$ et les définissons comme suit :

Définition 4.39. Soient $\mathfrak{teo}$ une théorie de sécurité et $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ l'ensemble des prédicats de $\tilde{\Sigma}^{Env}_{\mathfrak{teo}} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$ tels que $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}_i) = empty$. Une **contrainte $\mathfrak{teo}$-définissable** est une formule du second ordre de la forme

$$\exists \mathbf{q}_1, \ldots, \mathbf{q}_m, \left(\bigwedge_{i=i}^{n} \forall x_1, \ldots, x_{n_i},\ (\mathbf{p}_i(x_1, \ldots, x_{n_i}) \ \Leftrightarrow\ \varphi_i) \right)$$

où pour tout i, φ_i est une formule sans quantificateur (contenant éventuellement des égalités) sur la signature $\tilde{\Sigma}^{def}_{\mathfrak{teo}} = (\mathcal{S}, \mathcal{F}, \{\mathbf{p} \in \mathcal{P} \mid \mathcal{C}_{\mathfrak{teo}}(\mathbf{p}) \neq empty\})$ et l'ensemble de variables du second ordre $\mathcal{X}^{\mathcal{P}} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$. Par extension, on désigne par $\mathfrak{teo}$-définissable toute formule (éventuellement du premier ordre) équivalente à une contrainte de la forme présentée.

Exemple 4.40. Considérons par exemple une théorie de sécurité $\mathfrak{teo}$ contenant deux prédicats $\mathbf{p}_1 : \mathsf{s}$ et $\mathbf{p}_2 : \mathsf{s}$ tels que $\mathcal{C}_{\mathfrak{teo}}(\mathbf{p}_1) = \mathcal{C}_{\mathfrak{teo}}(\mathbf{p}_2) = empty$. La contrainte $\varphi = \forall x, \mathbf{p}_1(x) \Rightarrow \mathbf{p}_2(x)$ est $\mathfrak{teo}$-définissable car φ peut s'écrire sous la forme suivante :

$$\exists \mathbf{q}_1 : \mathsf{s}, \mathbf{q}_2 : \mathsf{s}, \begin{cases} & \forall x : \mathsf{s}, (\mathbf{p}_1(x) \ \Leftrightarrow\ \mathbf{q}_1(x) \wedge \mathbf{q}_2(x)) \\ \wedge & \forall x : \mathsf{s}, (\mathbf{p}_2(x) \ \Leftrightarrow\ \mathbf{q}_2(x)) \end{cases}$$

Définition 4.41. Soit $\mathfrak{teo}$ une théorie de sécurité. Une **requête administrative** sur $\mathfrak{teo}$ est une paire $req \parallel \varphi$ où φ est une contrainte $\mathfrak{teo}$-définissable et req un terme de $\mathcal{T}(\Sigma^{Act}_{\mathfrak{teo}}, \mathcal{X})$. L'ensemble des solutions de la requête $req \parallel \varphi$ dans une politique

de sécurité $\mathfrak{pol}$ (supposée terminante, cohérente et complète), noté $\mathcal{S}ol_{\mathfrak{pol}}(req \parallel \varphi)$ est donné par :

$$\left\{(\sigma, \psi, d) \mid \exists \eta \in \mathcal{E}nv(\mathfrak{teo}),\ \eta \models \varphi \wedge \psi, d \in \mathcal{D}_{\mathfrak{teo}},\ \sigma(req) \xrightarrow{*}_{\mathfrak{pol}} d\right\}$$

Les solutions indiquent la décision (d) prise par la politique en fonction des valeurs des variables de la requête (σ) et de conditions supplémentaires (ψ).

Exemple 4.42. Considérons la politique présentée dans l'exemple 4.18. On se pose la question de l'évaluation de la requête $req = \mathbf{take}(s, o, \mathbf{read})$ dans les environnements vérifiant :

$$\varphi = \forall s'{:}\mathsf{Subject}, o'{:}\mathsf{Object}, \begin{cases} s' = \mathbf{user}(i_s, l_s) \\ \wedge\ o' = \mathbf{object}(i_o, l_o) \\ \wedge\ \mathbf{accesses}(s', o', \mathbf{write}) \end{cases} \Rightarrow \mathbf{inf}(l_s, l_o)$$

en fonction des valeurs de s et de o. (φ, req) est une requête administrative sur $\mathfrak{teo}$ car φ est $\mathfrak{teo}$-définissable :

$$\exists \begin{cases} \mathbf{q_{sudo}} : \mathsf{Subject} \\ \mathbf{q_{acc}} : \mathsf{Subject} \times \mathsf{Object} \times \mathsf{Mode} \\ \mathbf{q_{blacklist}} : \mathsf{Subject} \\ \mathbf{q_{redlist}} : \mathsf{Subject} \end{cases}, \begin{cases} \forall x, \mathbf{sudo}(x) \ \Leftrightarrow\ \mathbf{q_{sudo}}(x) \\ \wedge\ \forall x, y, z, \mathbf{accesses}(x, y, z) \ \Leftrightarrow\ \varphi_{acc} \\ \wedge\ \forall x, \mathbf{blacklist}(x) \ \Leftrightarrow\ \mathbf{q_{blacklist}}(x) \\ \wedge\ \forall x, \mathbf{redlist}(x) \ \Leftrightarrow\ \mathbf{q_{redlist}}(x) \end{cases}$$

où φ_{acc} est la formule :

$$\mathbf{q_{acc}}(x, y, z) \wedge \forall i_s, l_s, i_o, l_o, \left(\begin{array}{c} x = \mathbf{user}(i_s, l_s) \\ \wedge\ y = \mathbf{object}(i_o, l_o) \\ \wedge\ z = \mathbf{write} \end{array}\right) \Rightarrow \mathbf{inf}(l_s, l_o)$$

L'ensemble des solutions de $req \parallel \varphi$ est donné par :

$$\{(\varnothing, \neg\mathbf{blacklist}(s), \mathbf{permit}), (\varnothing, \mathbf{blacklist}(s), \mathbf{deny})\}$$

Il indique que lorsque φ est vérifiée, l'accès en lecture à un objet est accordée à un utilisateur si et seulement si il n'est pas sur liste noire.

La question qui nous intéresse désormais est de savoir comment le cadre proposé permet de calculer les solutions d'une requête administrative. Tout d'abord, remarquons que la caractérisation des formules $\mathfrak{teo}$-définissables correspond aux contraintes qui peuvent être représentées sous la forme d'un terme avec variables sur la signature $\Sigma^{AR}_{\mathfrak{teo}}$.

Proposition 4.43. Soit $\mathfrak{teo}$ une théorie de sécurité et φ une contrainte $\mathfrak{teo}$-définissable. Il existe un système de réécriture à la racine $\mathcal{R}^{def}_{AR}$ sur une extension $\Sigma^{def}_{\mathfrak{teo}}$ de $\Sigma^{AR}_{\mathfrak{teo}}$ et un terme $\vartheta_{\mathfrak{teo}}(\varphi) \in \mathcal{T}(\Sigma^{def}_{\mathfrak{teo}}, \mathcal{X})$ tels que : $\eta \models \varphi$ ssi il existe une

substitution close σ telle que $\sigma(\vartheta_{\mathfrak{teo}}(\varphi)) \xrightarrow{*}_{\mathcal{R}_{AR}^{def}} \gamma_{\mathfrak{teo}}(\eta)$.

❝ *Démonstration.* Soit φ une contrainte $\mathfrak{teo}$-définissable exprimée sous la forme exposée dans la définition 4.39, c.-à-d. sous la forme :

$$\exists \mathbf{q}_1, \ldots, \mathbf{q}_m, \left(\bigwedge_{i=i}^{n} \forall x_1, \ldots, x_{n_i},\ (\mathbf{p}_i(x_1, \ldots, x_{n_i}) \ \Leftrightarrow \ \varphi_i) \right)$$

On associe à chaque prédicat $\mathbf{q}$ quantifié existentiellement une variable x_q de sorte $\mathsf{Facts_q} < \mathsf{Facts}$ et l'on augmente la signature $\Sigma_{\mathfrak{teo}}^{AR}$ des symboles suivants :

- $\mathbf{q} : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \to \mathsf{Fact_q}$,
- $\mathbf{nil_q} : \to \mathsf{Facts_q}$
- $\mathbf{cons_q} : \mathsf{Fact_q} \times \mathsf{Facts_q} \to \mathsf{Facts_q}$

Le système $\mathcal{R}_{AR}^{def}$ est composé des règles $C_i[\mathcal{R}_{AR}]$ pour tout $C_i = \langle x_1, \ldots, \mathbf{map_{p_i}}(\Box), \ldots, x_n \rangle$ ainsi que des règles $C_i'[map_{p_i}]$ où $C_i' = \langle\, x_1, \ldots, \Box, \ldots, x_n \,\rangle$ et map_{p_i} est l'ensemble de règles :

$$\begin{array}{rcl} \mathbf{map_{p_i}}(\mathbf{done}(x)) & \to & \mathbf{map_{p_i}^{aux}}(\mathbf{done}(x), [\,]) \\ \mathbf{map_{p_i}^{aux}}(\mathbf{done}(\langle\, x_1, \ldots, x_n \,\rangle :: q), l) & \to & \mathbf{map_{p_i}^{aux}}(\mathbf{done}(q), \mathbf{p_i}(x_1, \ldots, x_n) :: l) \\ \mathbf{map_{p_i}^{aux}}(\mathbf{done}([\,]), l) & \to & l \end{array}$$

Le terme $\langle\, \mathbf{map_{p_1}}(\gamma_{\mathfrak{teo}}(\varphi_1)), \ldots, \mathbf{map_{p_n}}(\gamma_{\mathfrak{teo}}(\varphi_n)) \,\rangle$ correspond alors à $\vartheta_{\mathfrak{teo}}(\varphi)$. ❞

En conséquence, la recherche des solutions d'une requête administrative peut être réduite à une union finie de problèmes d'accessibilité.

Proposition 4.44. Soit $\mathfrak{teo}$ une théorie de sécurité et $req \parallel \varphi$ une requête administrative sur $\mathfrak{teo}$. À toute solution (σ, ψ, d) de $\mathcal{S}ol_{\mathfrak{pol}}(req \parallel \varphi)$ correspond exactement une solution du problème d'accessibilité $\vartheta_{\mathfrak{teo}}(\varphi) \vdash req \xrightarrow{*}_{\mathcal{R}_{\mathfrak{pol}} \cup \mathcal{R}_{\mathfrak{teo}}^{AR}} d$.

Toute solution σ du problème d'accessibilité $\vartheta_{\mathfrak{teo}}(\varphi) \vdash req \xrightarrow{*}_{\mathcal{R}_{\mathfrak{pol}} \cup \mathcal{R}_{\mathfrak{teo}}^{AR}} d$ peut être partitionnée en deux substitutions $\sigma = \sigma_1 \cup \sigma_2$ telles que $\mathcal{D}om(\sigma_1) \subseteq \mathcal{V}ar(\vartheta_{\mathfrak{teo}}(\varphi))$ et $\mathcal{D}om(\sigma_2) \subseteq \mathcal{V}ar(req)$. σ_1 caractérise les conditions supplémentaires sous lesquelles $\sigma_2(req)$ est évalué en d.

4.4 Travaux reliés

Depuis une dizaines d'années, de nombreux travaux se sont attachés à établir des cadres formels de spécification pour l'analyse de politiques dynamiques.

Le premier article paru en ce sens est [Jajodia et al., 2001]. Dans ce papier, les auteurs présentent un cadre logique baptisé Flexible Authorization Framework (FAF) permettant d'exprimer et de mettre en oeuvre des politiques de contrôle d'accès multiples. Dans FAF, les règles sont spécifiées dans un style à la Prolog

[Sterling and Shapiro, 1994]. Le cadre inclus notamment la notion d'historique mémorisant les accès effectués. En ce sens, ce travail constitue une première étape vers une définition plus générique de la notion de politiques dynamiques. Dans la même veine, les travaux présentés dans [Becker and Sewell, 2004] proposent un langage et un système basé sur les rôles pour exprimer les politiques. Le langage, Cassandra, supporte le contrôle d'accès basé sur les certificats et se fonde sur une extension de Datalog intégrant la notion de contraintes. Les faits évoluent pour modéliser les effets de bords induits par l'exécution des actions. Les politiques sont vues comme des déclarations sur les données des requêtes et sur les relations capturant les informations recueillies par l'application. Une des principales limitations de ces travaux est le caractère spécifique de la démarche aux politiques basés sur la notion de rôles.

Dans [Dougherty et al., 2006], les auteurs préconisent également une séparation de la politique de sécurité, définie statiquement, du comportement dynamique du programme et proposent une représentation de la politique dynamique induite par ces deux éléments (règles statiques et programme). La notion d'environnement est introduite sous la forme d'un ensemble de faits caractérisant le système. L'exécution d'une action est associée à l'ajout ou à la suppression de faits à partir de l'environnement duquel l'action a été invoquée. Les politiques de sécurité sont modélisées par des programmes logiques Datalog de la forme suivante :

$$\begin{aligned} \textbf{permit}(x_1,\ldots,x_n) &\leftarrow p_1(x_1^1,\ldots,x_{n_1}^1)\wedge\ldots\wedge p_k(x_1^k,\ldots,x_{n_k}^k)\\ \textbf{deny}(x_1,\ldots,x_n) &\leftarrow p_1(x_1^1,\ldots,x_{n_1}^1)\wedge\ldots\wedge p_k(x_1^k,\ldots,x_{n_k}^k) \end{aligned}$$

Les auteurs s'attachent à montrer que ce cadre permet d'analyser formellement les propriétés de sûreté et de disponibilité. Cependant, l'usage de Datalog limite le pouvoir d'expression du cadre proposé.

Dans [Becker, 2009], Becker propose un langage de spécification appelé DynPAL permettant de vérifier les propriétés de sûreté sur des politiques dynamiques possédant un nombre non borné d'objets. Le papier propose deux méthodes pour raisonner sur l'accessibilité et les invariants des politiques. Pour traiter le problème de l'accessibilité, la politique est traduite dans le langage PDDL [McDermott et al., 1998]. Les propriétés de sûreté sont vérifiées en utilisant un prouveur de théorèmes du premier ordre et réduit la vérification de l'hypothèse d'invariance à un problème de validité d'une formule du premier ordre.

Dans [Becker and Nanz, 2010], un langage baptisé SMP est présenté. Destiné à décrire les politiques avec effets de bords, le langage est fondé sur la logique transactionnelle. Cette logique est une extension de la logique des prédicats comportant à la fois une sémantique déclarative et procédurale décrivant les changements d'états d'une base de données dynamique. Le fragment retenu pour la description des politiques est basé sur Datalog. Il souffre donc d'une expressivité limitée.

4.5 Synthèse

Nous avons présenté dans ce chapitre un cadre permettant de spécifier séparément les règles d'une politique de sécurité du système sur lequel elle s'applique.

Les états du systèmes sont vus comme des interprétations logiques finies (ensembles finis de faits) et les transitions sont décrites au moyen de règles de mises

à jour des interprétations. Les politiques sont exprimées au moyen de règles de réécriture contraintes par des formules du premier ordre. Nous avons ensuite défini la notion de système sécurisé, matérialisant le résultat de l'application d'une politique de sécurité sur un système.

Nous avons proposé un langage dont la sémantique opérationnelle des spécifications est donnée au moyen de systèmes de réécriture. La mise en place d'un langage dédié permet d'imposer des restrictions syntaxiques complexes sur la logique sous-jacente (les systèmes de réécriture) de façon transparente pour l'utilisateur. En ce sens, notre approche se distingue fortement de la plupart des approches formelles proposées pour la spécification des politiques dans la littérature. La forme particulière des systèmes obtenus permet d'utiliser les outils et techniques existants pour analyser les politiques ainsi spécifiées : étude des propriétés (cohérence, terminaison, complétude), du problème de l'administration, comparaison des politiques et analyse par requêtage.

Une partie des travaux présentés dans ce chapitre ont été présentés dans [Bourdier et al., 2010b, Bourdier et al., 2010c, Bourdier et al., 2011].

5 Les politiques de sécurité dans les réseaux informatiques

Sommaire

5.1 Firewalls : une introduction informelle 107
5.2 Spécification formelle de firewalls 111
5.2.1 Vocabulaire et définition 111
5.2.2 Sémantique basée sur les automates d'arbres 113
5.3 Applications 118
5.3.1 Propriétés 119
5.3.2 Analyse structurelle 120
5.3.3 Analyse par requêtage 125
5.4 Composition de firewalls 126
5.4.1 Topologies de réseaux 126
5.4.2 Politiques de sécurité réseau 128
5.4.3 Stratégie de routage 130
5.4.4 Sémantique par réécriture 131
5.4.5 Propriétés et analyse 140
5.5 Travaux reliés 145
5.6 Synthèse 146

Après la seconde guerre mondiale, l'informatique restait l'apanage de grandes entreprises qui pouvaient se permettre l'acquisition de quelques ordinateurs. La communication de station à station s'effectuait alors au moyen de supports amovibles. La taille et la complexité des organisations ayant crû significativement,

la volonté de faciliter la communication au sein d'une même structure s'est vite ressentie, entraînant la nécessité d'envisager un autre moyen de communication plus efficace. C'est ainsi que le premier mode de communication dit de « poste-à-poste » vit le jour dans les années soixante grâce aux recherches menées par le consortium DIX (Digital, Intel, Xerox). Il devenait alors possible de communiquer directement avec un ordinateur central (mainframe) à partir de sa station de travail. Ce n'est que vers la fin de la décénnie et à l'aube des années soixante-dix que le développement des véritables réseaux locaux (ou LAN [1]) a commencé, avec l'idée de ne plus nécessairement passer par un ordinateur principal. La décentralisation de l'information s'amorçait alors, tandis que l'idée d'une communication directe vers le monde extérieur faisait son chemin. C'est en 1969 que le projet ARPA initié par le département de la défense des Etats-Unis s'est donné pour objectif de relier quatre centres universitaires (l'université de Californie à Los Angeles, l'institut de recherche de Stanford, l'université de Californie à Santa Barbara et l'université de l'Utah). Le réseau ainsi créé évolua jusqu'à intégrer vingt-trois centres universitaires en 1971 puis plus de cent en 1977. Les réseaux étaient nés, les possibilités de communications décuplées et par la même la nécessité de protéger les accès à des machines (et des informations) désormais vulnérables.

C'est le 2 novembre 1988, onze années plus tard, que se produisit l'inévitable. Dans un communiqué de Peter Yee, chercheur à la NASA, adressé à la communauté scientifique travaillant sur l'Internet, on pouvait lire : « Nous sommes actuellement victimes d'une attaque d'un virus internet ! Il a frappé Berkeley, UC San Diego, Lawrence Livermore, Stanford, ainsi que le centre de recherche de la NASA » [Avolio, 1999]. La communauté scientifique et technique à l'initiative de cette formidable avancée technologique qu'est l'Internet réalisa alors qu'en permettant à des individus de communiquer librement, ils permettaient aussi à des individus malintentionnés de s'introduire dans leurs systèmes. On avait troqué un environnement sûr et confiné pour un environnement ouvert et potentiellement hostile. D'autres attaques se succédèrent, faisant les choux gras de la presse et annonçant la nécessité absolue de sécuriser les réseaux de communication. Les politiques de sécurité s'imposèrent alors et les mécanismes pour les mettre en oeuvre, les fameux firewalls, apparurent.

Développer une politique de sécurité, nous l'avons vu, n'est pas chose aisée mais est une étape indispendable pour se prémunir des attaques d'individus malintentionnés. Mettre en oeuvre une politique de sécurité consiste à garantir l'application des règles de la politique par les mécanismes opérationnels fournis par le système. Les firewalls (ou pare-feux) sont présentés depuis toujours comme le principal moyen de mettre en application une politique de sécurité dans un réseau informatique. Nous proposons dans ce chapitre un cadre pour analyser formellement la mise en oeuvre de politiques de sécurité réseau exprimées comme des combinaisons de firewalls. Dans un premier temps, nous présenterons informellement le fonctionnement des firewalls. Nous poursuivrons ensuite en caractérisant formellement chacun des éléments constitutifs de ces derniers. Nous montrerons en quoi cette caractérisation permet de démontrer la décidabilité d'une large classe de problèmes. Nous analyserons ensuite la composition de firewalls. Une partie des travaux présentés dans ce chapitre a été publiée dans [Bourdier, 2011b] et [Bourdier and Cirstea, 2011].

1. Local Area Network

5.1 Firewalls : une introduction informelle

Dans cette section, nous proposons d'envisager les notions nécessaires à la compréhension de ce chapitre en dehors de considérations trop techniques, garantissant ainsi une présentation simple et compréhensible des non spécialistes. Si l'exposé de ces éléments est réalisé dans un souci de simplification mais aussi de fidélité à la réalité, il n'échappera pas aux spécialistes de la question qu'un certain nombre de subtilités sont passées sous silence afin d'éviter toute digression nuisant au développement qui nous intéresse.

Dans la suite, nous désignerons par « hôte » toute entité connectée au réseau. Les notions de sujets, d'objets et d'action n'auront pas cours dans un premier temps. Une interprétation de ces concepts sera donnée dans la dernière section de ce chapitre. Dans un réseau de communication, lorsqu'un hôte souhaite transmettre un message à un autre hôte, les données du message sont découpées et encapsulées dans une séquence de paquets. Chaque paquet contient les données à transmettre ainsi que les informations nécessaires pour acheminer le paquet jusqu'à son destinataire. L'ensemble de ces informations supplémentaires est appelé en-tête.

L'objectif des firewalls est de contrôler et de filtrer le traffic transitant entre différents sous-réseaux. Rappelons qu'un sous-réseau est une subdivision logique du réseau caractérisé par un ensemble d'adresses IP. Les adresses (IP) sont un moyen d'identifier les éléments connectés au réseau et ne peuvent posséder au maximum qu'une occurence au sein d'un même sous-réseau. Les adresses (IPv4 [2]) sont des séquences de 32 bits scindées en deux parties : un préfixe identifiant le sous-réseau auquel appartient l'hôte et un suffixe, ou « numéro d'hôte », permettant d'identifier un hôte au sein de son sous-réseau.

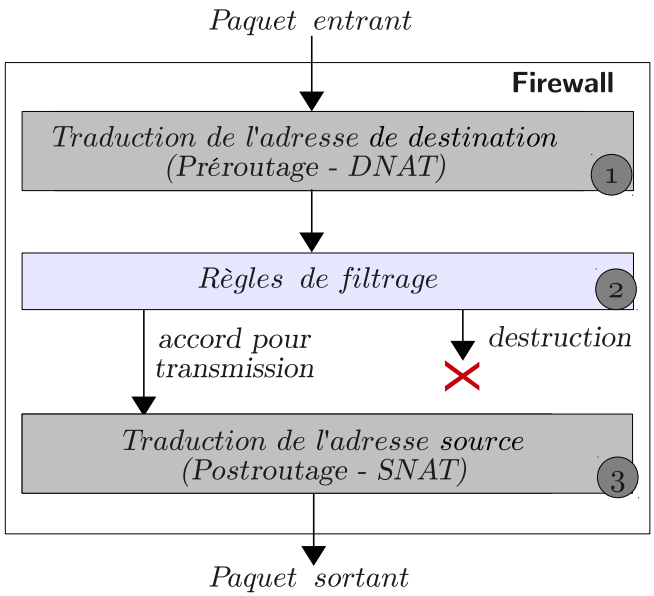
Un firewall est donc une application qui contrôle la transmission des paquets qui le traverse à partir des deux fonctionnalités suivantes :

- *le filtrage de paquet*, qui consiste à inspecter l'en-tête de chaque paquet pour soit le laisser poursuivre son chemin, soit le détruire et
- *la traduction d'adresse réseau*, qui consiste à modifier dans l'en-tête les informations relatives aux adresses de l'émetteur et du destinataire du paquet.

Ainsi, les firewalls inspectent les paquets entrants et acceptent ou refusent de les transmettre en basant leur décision sur une liste de règles. Chaque règle associe un motif décrivant un ensemble de paquets à une décision particulière. Les critères les plus souvent utilisés lors de l'inspection des paquets sont [Conoboy and Fictner, 2002, Russell, 2002] l'adresse de l'émetteur du paquet (appelée adresse source), celle de son destinataire (appelée adresse de destination), le protocole utilisé et, lorsque le protocole est TCP ou UDP, un numéro de port. Le processus de traduction d'adresses, ou NAT (pour network address translation), effectue une réécriture de l'adresse source du paquet avant le processus de filtrage (cette partie du processus est appelée DNAT [3], ou pré-routage) et, à l'issue du filtrage, réécrit l'adresse de destination du paquet (cette partie du processus est appellée SNAT [4], ou post-routage). Le

2. IPv4 est la version d'Internet Protocol actuellement utilisée dont un des objectifs est de fournir un service d'adressage unique pour l'ensemble des éléments connectés en réseau.
3. DNAT : Destination Network Address Translation
4. SNAT : Source Network Address Translation

diagramme suivant résume le fonctionnement général d'un firewall :



À chaque étape (1, 2 and 3), le paquet entrant est comparé à une liste de règles et l'action (traduire l'adresse réseau de destination, accepter ou refuser la transmission et traduire l'adresse réseau source) associée à la première règle reconnaissant le paquet est effectuée. Il existe plusieurs types de traduction d'adresses réseau. Nous nous focalisons sur les types de traductions suivantes :

- Le processus de traduction le plus simple est de type one-to-one (un pour un) aussi appelé NAT basique [Srisuresh and Holdrege, 1999] et consiste à associer statiquement une adresse IP avec une autre adresse IP :

$$\begin{cases} ip_1 \leftrightarrow ip'_1 \\ ip_2 \leftrightarrow ip'_2 \\ \vdots \\ ip_n \leftrightarrow ip'_n \end{cases}$$

- Une autre méthode (masquerading) consiste à associer un ensemble d'adresses IP avec une seule autre adresse IP. Dans ce cas, ce sont les numéros de ports qui permettent d'identifier la traduction en place : le numéro du port source va être modifié dynamiquement et servira à identifier l'adresse IP d'origine. On ne connaît donc pas statiquement les associations qui seront effectuées, le port étant choisi dynamiquement dans un intervalle donné.

$$(ip^? \in [ip_1, ip_n], port^?) \leftrightarrow (ip', port'^?)$$

  $ip^? \in [ip_1, ip_n]$ signifie que $ip^?$ est une adresse IP quelconque appartenant à l'intervalle $[ip_1, ip_n]$.
- Les techniques appelées port forwarding et port mapping, également de type one-to-one, consistent à rediriger un paquet vers une machine précise en fonction du port de destination de ce paquet. Toutes les informations sont connues

statiquement :

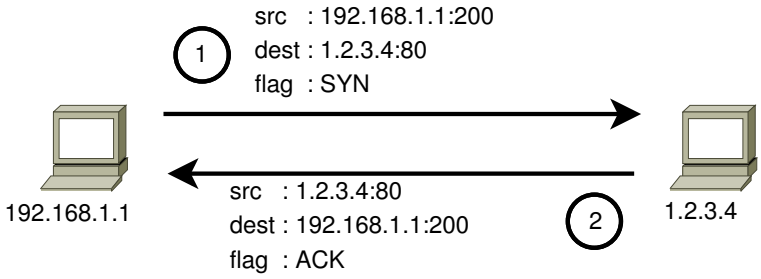
$$(ip, port) \leftrightarrow (ip', port')$$

(dans le cas du port forwarding, $port = port'$)

Notation CIDR

La notation CIDR [Fuller and Li, 2006] est une forme d'écriture des adresses de sous-réseau sous la forme d'un couple séparé par un « slash » dont le premier élément indique l'adresse en notation décimale pointée de la plus petite adresse IP appartenant au sous-réseau et le second indique le nombre de bits de l'adresse dédiés au préfixe du sous-réseau. Ainsi, 192.168.1.1/25 signifie que les 25 premiers bits de l'adresse correspondent à l'adresse du sous-réseau, et que les 32-25=7 bits restant identifient l'adresse de l'ordinateur hôte à l'intérieur du sous-réseau. On utilisera également la notation $ip{:}pt$ pour représenter le couple composé de l'adresse IP ip et du port pt.

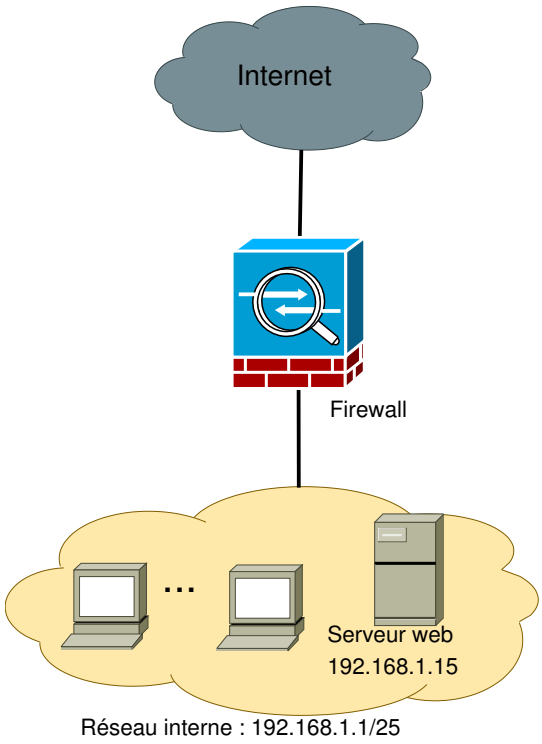
Dans la suite, nous supposerons que seul le protocole TCP est utilisé et ne considèrerons que les champs relatifs aux adresses et aux ports. TCP est un protocole dit « orienté connection ». Cela signifie que lorsqu'un paquet est envoyé, il comporte une information indiquant s'il est le premier paquet d'une communication donnée ou s'il fait suite à un paquet préalablement reçu. Exemple :



L'hôte dont l'adresse est 192.168.1.1 a initié une communication avec l'hôte dont l'adresse est 1.2.3.4. Le paquet correspondant contient alors le flag SYN (pour SYNchronisation). Les paquets faisant suite à ce premier échange contiendront tous le flag ACK (ACKnowledgment). Dans la suite, nous ne préciserons pas les drapeaux de connection (SYN et ACK) et considèrerons que dans le cas des règles de filtrage ne sont spécifiées que les autorisations correspondant à des initialisations de connection. Les paquets « réponses » correspondants seront implicitement acceptés. Dans le cas de la traduction d'adresse, les règles de masquerading déterminent à l'initialisation de la connection le port qui sera utilisé pour identifier l'hôte dont l'adresse est « cachée » et les paquets faisant suite à cette initialisation utiliseront la correspondance ainsi établie. Nous considérerons également que l'intervalle de ports utilisé pour le masquerading correpond aux ports dont l'écriture binaire commence par 11 (puisque les ports sont définis par 16 bits, il s'agit de l'intervalle $[49152, 65535]$, parfois appelé intervalle des ports dynamiques). En somme, les règles de filtrage sont supposées s'appliquer aux paquets contenant le drapeau SYN et est supposée présente une règle par défaut pour les paquets contenant le drapeau ACK. Les règles SNAT correspondant à une étape de masquerading seront supposées être associées au drapeau SYN et leurs homologues DNAT seront implicitement spécifiés et associés au drapeau ACK (et ne s'appliquent donc pas pour un paquet initialisant une connection).

> *Exemple 5.1.* Considérons une architecture de réseau simple comportant uniquement un réseau interne séparé d'Internet par un firewall. La plage d'adresses IP attribuée au réseau interne est 192.168.1.1/25 (notation CIDR). Les adresses des hôtes du réseau interne ne doivent pas être visibles sur Internet, c'est pourquoi l'on dote le firewall de règles de traduction d'adresses faisant la correspondance entre les adresses « internes » 192.168.1.1/25 et leur adresse publique 152.81.13.1 (masquerading). La politique de filtrage choisie consiste à permettre aux paquets provenant d'Internet d'accéder au serveur

WEB uniquement. Pour cela, on redirige les requêtes à destination du port 80 du firewall (152.81.13.1) sur le port 80 du serveur (192.168.1.15). Les autres paquets provenant d'Internet sont jetés. Les hôtes du réseau interne sont, quant à eux, uniquement autorisés à consulter leurs emails avec le protocole SMTP (le protocole de communication SMTP est réalisé par l'utilisation du port 25 du protocole de transport TCP).



Les règles de filtrage du firewall sont donc les suivantes :

IP address src	*IP address dest*	*Port src*	*Port dst*	*Decision*
$*$	192.168.1.15	$*$	80	**accept**
192.168.1.15	$*$	80	$*$	**accept**
192.168.1.1/25	$*$	$*$	25	**accept**
$*$	$*$	$*$	$*$	**drop**

et celles de traduction d'adresses sont les suivantes :

SNAT/DNAT	*IP Address*	*Port*	*new IP Address*	*New Port*
SNAT	192.168.1.15	80	152.81.13.1	80
SNAT	192.168.1.1/25	$*$	152.81.13.1	*dyn.*
DNAT	152.81.13.1	80	192.168.1.15	80

Ainsi, un paquet en provenance d'un hôte identifié sur internet par l'adresse IP 252.45.32.114 à destination du port 80 de l'adresse 152.81.13.1 sera traduit (DNAT) en paquet à destination de 192.168.1.15 (port 80) et sera ainsi délivré à son destinataire (première règle de filtrage). En revanche, si ce même paquet

était à destination de l'adresse 152.81.13.1 sur le port 49153, deux cas de figure se présentent. Soit ce paquet fait réponse à un paquet précédent émis par un hôte et dans ce cas l'adresse de destination (152.81.13.1 :3021) est traduite par l'adresse et le port utilisés par l'hôte lors de l'initialisation de la connection. Soit ce paquet est une initialisation de connection et dans ce cas, aucune règle DNAT ne s'applique et alors le paquet est jeté (dernière règle de filtrage).

5.2 Spécification formelle de firewalls

5.2.1 Vocabulaire et définition

Paquets et sous-réseaux

Dans l'objectif de donner une sémantique formelle à chaque composant d'un firewall, nous avons besoin de définir le vocabulaire à partir duquel nous décrirons chacun des objets de notre étude (IP, paquets, ...). Comme dans le reste de ce manuscrit, nous fondons notre travail sur une représentation symbolique de chacun des éléments étudiés. Pour conserver un maximum de lisibilité, nous considèrerons par la suite que les éléments filtrés par les firewalls sont réduits aux adresses et au ports. Les autres informations, telles que les protocoles, les flags TCP, les états,... peuvent être considérées sans difficulté majeure. La représentation symbolique retenue des éléments précédemment décrits est basée sur la signature Σ suivante :

$$\begin{array}{lll} \mathbf{0}, \mathbf{1} & : & \mathsf{Binary} \rightarrow \mathsf{Binary} \\ \# & : & \rightarrow \mathsf{Binary} \\ \mathbf{ip} & : & \mathsf{Binary} \rightarrow \mathsf{IP} \\ \mathbf{port} & : & \mathsf{Binary} \rightarrow \mathsf{Port} \\ \mathbf{packet} & : & \mathsf{IP} \times \mathsf{Port} \times \mathsf{IP} \times \mathsf{Port} \rightarrow \mathsf{Packet} \end{array}$$

Les adresses IP sont décrites par des termes clos de sorte IP correspondant à leur représentation binaire. Par exemple, l'adresse IP 192.168.1.1 est symboliquement représentée par le terme $\mathbf{ip}(\mathbf{11000000\,10101000\,00000001\,00000001}(\#))$ (nous utiliserons la notation préfixée pour les symboles de fonction **0** et **1**) étant entendu que la notation binaire de 192 est 11000000 et que celle de 168 est 10101000. Nous procédons de la même façon pour les ports (modulo le remplacement de **ip** par **port**). Enfin, les paquets sont représentés par des termes de sorte Packet. Pour des raisons de lisibilité, nous utiliserons dans ce papier la notation décimale pointée pour représenter les adresses et la notation décimale pour les ports. Par exemple, $\mathbf{packet}(\mathbf{ip}(\mathit{192.168.1.1}), \mathbf{port}(\mathit{80}), \mathbf{ip}(\mathit{172.20.3.1}), \mathbf{port}(\mathit{80}))$ doit s'entendre comme le terme $\mathbf{packet}(\mathbf{ip}(t_s), \mathbf{port}(t_p), \mathbf{ip}(t_d), \mathbf{port}(t_p))$ où

$$\begin{cases} t_s = \mathbf{10000000\ 10000000\ 00010101\ 00000011}(\#) \\ t_d = \mathbf{10000000\ 11000000\ 00101000\ 00110101}(\#) \\ \text{et } t_p = \mathbf{0000101000000000}(\#) \end{cases}$$

Par convenance, nous supposerons que tout terme de sorte IP (quelle que soit sa taille) représente une adresse IP. Il représentera une adresse IPv4 (resp. IPv6) ssi il contient exactement 32 (resp. 128) symboles **0** et **1**.

Une autre notion qu'il est important de pouvoir caractériser symboliquement est celle de sous-réseau. Rappelons qu'un sous-réseau est une subdivision logique d'un réseau IP caractérisée par l'ensemble des adresses IP des hôtes qu'il contient. Les adresses IPv4 sont composées de deux parties : un préfixe caractérisant le sous-réseau auquel l'adresse appartient et un suffixe correspondant au numéro de l'hôte dans le sous-réseau. Plus précisément, lorsque l'on définit un sous-réseau, on définit un nombre $n < max$ de bits (où max est 32 dans le cas des adresses IPv4 et 128 dans le cas IPv6) ainsi qu'une séquence de n bits qui caractérise le préfixe du sous-réseau. Les $max - n$ bits restant servent à identifier l'hôte dans le sous-réseau. Par exemple, le réseau (notation CIDR) 192.168.5.64/26 représente l'ensemble des hôtes dont la représentation binaire des adresses IP possède les 26 mêmes premiers bits que la représentation binaire de 192.168.5.64. Puisque la seule relation dont nous disposons pour « comparer » les termes est la relation de subsomption, la relation d'inclusion entre sous-réseaux (et d'appartenance d'une adresse à un sous-réseau) doit se caractériser par une relation de subsomption (et de filtrage) entre leurs représentations symboliques. La représentation symbolique des sous-réseaux devient alors naturelle compte-tenu du fait que les adresses IP appartenant à un sous-réseau sont caractérisées par un préfixe commun. Tout sous-réseau sera donc représenté par un terme linéaire de sorte IP (et réciproquement tout terme linéaire de sorte IP représente un sous-réseau). Par exemple, le sous-réseau 192.128.0.1/10, caractérisé par le préfixe 1100000010, sera représenté par le terme $\mathbf{ip}(\mathbf{11000000\,10}(x))$. Ainsi, l'ensemble des adresses des hôtes de ce sous-réseau correspond à l'ensemble des instances closes de ce terme. Pour des raisons de lisibilité, nous emploierons la notation CIDR pour représenter les sous-réseaux, même symboliquement. Par exemple, la notation $\mathbf{ip}(192.128.0.1/10[x])$ sera utilisée en lieu et place du terme $\mathbf{ip}(\mathbf{11000000\,10}(x))$.

Spécification des règles

Pour spécifier les règles de firewalls, nous ajoutons à la signature Σ les deux symboles suivants :

$$\mathbf{accept}, \mathbf{drop} \quad : \quad \rightarrow \mathsf{Decision}$$

La définition formelle d'un firewall peut alors s'exprimer de la façon suivante :

Définition 5.2. Un **firewall** $\mathfrak{fw}$ est la donnée :

- d'un ensemble ordonné (ordre noté $<_{\mathfrak{fw}}$) $\mathrm{Filter}_{\mathfrak{fw}}$ de règles de la forme $lhs \rightarrowtriangle rhs$ où lhs est un terme linéaire de sorte Packet et rhs un terme clos de sorte $\mathsf{Decision}$;
- de deux ensembles ordonnés (ordre également noté $<_{\mathfrak{fw}}$) $\mathrm{Pre}_{\mathfrak{fw}}$ et $\mathrm{Post}_{\mathfrak{fw}}$ de règles $lhs \rightarrowtriangle rhs$ de l'une des formes suivantes :
  - (i) lhs et rhs sont deux termes clos de sorte IP,
  - (ii) lhs et rhs sont deux termes clos de sorte Port,
  - (iii) lhs et rhs sont deux couples de termes clos de sorte $\mathsf{IP} \times \mathsf{Port}$,
  - (iv) lhs est un terme de sorte IP contenant une variable et rhs est un terme clos de sorte IP (dans le cas de $\mathrm{Post}_{\mathfrak{fw}}$ uniquement).

$(i), (ii), (iii)$ correspondent aux règles NAT de type one-to-one et le cas (iv) correspond au masquerading (many-to-one).

Exemple 5.3. Le firewall décrit dans l'exemple 5.1 peut être spécifié de la façon suivante :

- $\text{Filter}_{\mathfrak{fw}}$ contient les règles (ordonnées) :

$$\begin{cases} \mathbf{packet}(x, y, \mathbf{ip}(192.168.1.15), \mathbf{port}(80)) & \twoheadrightarrow \ \mathbf{accept} \\ \mathbf{packet}(\mathbf{ip}(192.168.1.15), \mathbf{port}(80), x, y) & \twoheadrightarrow \ \mathbf{accept} \\ \mathbf{packet}(\mathbf{ip}(192.168.1.1/25[x]), y, z, \mathbf{port}(25)) & \twoheadrightarrow \ \mathbf{accept} \\ \mathbf{packet}(x, y, z, u) & \twoheadrightarrow \ \mathbf{drop} \end{cases}$$

- $\text{Pre}_{\mathfrak{fw}}$ contient la règle :

$$\{ \ \mathbf{ip}(152.81.13.1), \mathbf{port}(80) \twoheadrightarrow \mathbf{ip}(192.168.1.15), \mathbf{port}(80)$$

- et $\text{Post}_{\mathfrak{fw}}$ contient les règles :

$$\begin{cases} \mathbf{ip}(192.168.1.15), \mathbf{port}(80) & \twoheadrightarrow \ \mathbf{ip}(152.81.13.1), \mathbf{port}(80) \\ \mathbf{ip}(192.168.1.1/25[y]) & \twoheadrightarrow \ \mathbf{ip}(152.81.13.1) \end{cases}$$

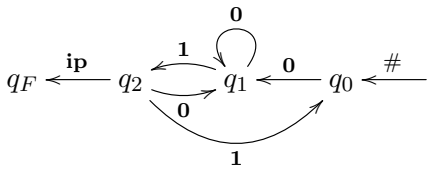
5.2.2 Sémantique basée sur les automates d'arbres

Dans cette section nous allons montrer que les analyses usuelles peuvent être effectuées de façon automatique en munissant les firewalls d'une sémantique basée sur les automates d'arbres. Pour cela, il est nécessaire de montrer que l'ensemble des paquets caractérisés par les règles des firewalls sont des ensembles réguliers puis de montrer que la combinaison des règles (dans le cas du filtrage et de la traduction d'adresse) préservent la régularité.

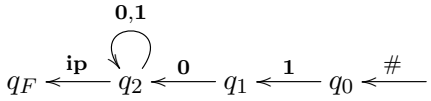
Proposition 5.4. L'ensemble des représentations symboliques des adresses IP appartenant à un sous-réseau donné est un ensemble régulier.

Comme nous l'avons déjà évoqué, l'ensemble des adresses IP d'un sous-réseau donné correspond à l'ensemble des instances closes du terme linéaire représentant ce sous-réseau. De ce fait, cet ensemble est effectivement régulier. Dans un soucis d'efficacité, nous nous proposons de ne pas représenter l'ensemble des adresses IP d'un sous-réseau par un automate reconnaissant les instances closes du terme correspondant, et optons pour une représentation légèrement différente. Pour réduire la taille des automates considérés, les adresses IP seront représentées par le terme correspondant à la représentation inversée de leur notation binaire. L'automate correspondant est non seulement plus simple à construire de façon automatique mais possède également des propriétés particulières permettant d'effectuer les opérations booléeennes avec une complexité linéaire. À titre d'exemple, considérons le sous-réseau 128.0.0.1/2 caractérisé par le préfixe 10. L'automate déterministe minimal reconnaissant l'ensemble des adresses de ce sous-réseau est le suivant (pour permettre

au lecteur d'appréhender plus facilement les automates manipulés, ces derniers sont représentés par un graphe sagittal, ce qui est rendu possible par l'usage de symboles d'arité au plus 1) :



où q_F est l'unique état final. L'automate déterministe minimal représentant les notations inversées des mêmes adresses est le suivant :



Cela vient du fait que les automates d'arbres que nous utilisons sont de type « bottom-up » et qu'un automate de mots correspond à un automate de type « top-down ». Étant entendu que les représentations des adresses IP sont désormais inversées, donnons la procédure générale pour construire l'automate d'arbres reconnaissant les adresses d'un sous-réseau. Soit un sous-réseau caractérisé par le préfixe $b_1, \ldots, b_n$. L'automate minimal déterministe reconnaissant l'ensemble des adresses IP appartenant à ce sous-réseau est donné par :

$$\left\{ \begin{array}{rcl} \mathbf{ip}(q_n) & \to & q_F \\ \mathbf{0}(q_n) & \to & q_n \\ \mathbf{1}(q_n) & \to & q_n \end{array} \right\} \cup \{b_i(q_{i-1}) \to q_i \mid i = 1, \ldots, n\} \cup \{\# \to q_0\}$$

Notons par ailleurs que si l'on note $E[n]$ le sous-ensemble de termes d'un ensemble quelconque de termes E qui sont de longueurs n, alors, pour toute opération booléenne $\oplus$, on a $E[n] \oplus F[n] = (E \oplus F)[n]$. Autrement dit, il n'est pas nécessaire de restreindre les ensembles de termes aux termes « valides » avant d'effectuer des calculs, la restriction pouvant se faire a posteriori. Nous pouvons donc considérer dans la suite les automates reconnaissant des adresses IP de longueur arbitraire.

Règles de filtrage

Commençons par donner la sémantique des règles de filtrage. Puisque $\mathcal{T}(\Sigma)_{\mathsf{Packet}}$ représente l'ensemble de tous les paquets, alors à toute règle de filtrage $r = lhs \rightarrowtriangle rhs \in \mathrm{Filter}_{\mathfrak{fw}}$ est associé un sous-ensemble de $\mathcal{T}(\Sigma)_{\mathsf{Packet}}$ correspondant à l'ensemble des paquets vérifiant toutes les conditions spécifiées par lhs. Pour tout paquet $p \in \mathcal{T}(\Sigma)_{\mathsf{Packet}}$ et toute règle de filtrage r, on notera $r \ll_{\mathrm{Filter}} p$ pour indiquer que le paquet p filtre la règle r (*i.e.* vérifie toutes les conditions de r). Puisque lhs est un terme linéaire, cet ensemble est l'ensemble des instances closes de lhs (modulo la représentation inversée des adresses). On obtient donc le résultat suivant :

Proposition 5.5. Pour toute règle de filtrage r, l'ensemble $\ll_{\mathrm{Filter}}(r) = \{p \in \mathcal{T}(\Sigma)_{\mathsf{Packet}} \mid r \ll_{\mathrm{Filter}} p\}$ est effectivement régulier.

Ceci étant, puisque les règles de filtrages sont évaluées en fonction de l'ordre $<_{\mathfrak{fw}}$ (seule la première règle filtrée est appliquée), alors, l'ensemble des paquets qui filtrent effectivement une règle dépend de l'ensemble des règles plus prioritaires. Pour tout firewall $\mathfrak{fw}$ et pour toute règle de filtrage r de $\mathfrak{fw}$, nous définissons la relation $\ll^{\mathfrak{fw}}_{\text{Filter}}$ de la façon suivante : $r \ll^{\mathfrak{fw}}_{\text{Filter}} p$ ssi $r \ll_{\text{Filter}} p$ et s'il n'existe aucune règle plus prioritaire $r' <_{\mathfrak{fw}} r$ dans $\text{Filter}_{\mathfrak{fw}}$ telle que $r' \ll_{\text{Filter}} p$.

Proposition 5.6. Pour tout firewall $\mathfrak{fw}$ et pour toute règle de filtrage $r \in \text{Filter}_{\mathfrak{fw}}$, l'ensemble $\ll^{\mathfrak{fw}}_{\text{Filter}}(r) = \{p \in \mathcal{T}(\Sigma)_{\mathsf{Packet}} \mid r \ll^{\mathfrak{fw}}_{\text{Filter}} p\}$ est effectivement régulier.

Nous avons vu que l'objectif d'une règle de filtrage est d'associer un certain ensemble de paquets à une décision (**accept** ou **drop**). On peut ainsi voir toute règle de filtrage r comme une fonction partielle $\overset{r}{\longmapsto}_{\text{Filter}}$ associant tout paquet p tel que $r \ll_{\text{Filter}} p$ à un élément de $\{\mathbf{accept}, \mathbf{drop}\}$. De plus, comme précédemment, nous définissons une fonction partielle $\overset{r}{\longmapsto}^{\mathfrak{fw}}_{\text{Filter}}$ associant chaque paquet p tel que $r \ll^{\mathfrak{fw}}_{\text{Filter}} p$ à la décision (**accept** ou **drop**) associée à r.

Proposition 5.7. Pour tout firewall $\mathfrak{fw}$ et pour toute règle de filtrage $r \in \text{Filter}_{\mathfrak{fw}}$, les fonctions partielles $\overset{r}{\longmapsto}_{\text{Filter}}$ et $\overset{r}{\longmapsto}^{\mathfrak{fw}}_{\text{Filter}}$ sont effectivement fortement régulières (en tant que relations, puisqu'une fonction est une relation particulière).

Enfin, on fait correspondre à tout firewall $\mathfrak{fw}$ une « sémantique de filtrage » sous la forme d'une fonction associant tout paquet à une décision. Cette fonction est, par définition :

$$\longmapsto^{\mathfrak{fw}}_{\text{Filter}} = \bigcup_{r \in \text{Filter}_{\mathfrak{fw}}} \overset{r}{\longmapsto}^{\mathfrak{fw}}_{\text{Filter}}$$

(où les fonctions $\overset{r}{\longmapsto}^{\mathfrak{fw}}_{\text{Filter}}$ sont vues comme des relations dont les domaines sont deux à deux disjoints). On obtient la proposition suivante :

Proposition 5.8. Pour tout firewall $\mathfrak{fw}$, $\longmapsto^{\mathfrak{fw}}_{\text{Filter}}$ est effectivement fortement régulière.

Règles de traduction d'adresses

L'objectif des règles de traduction d'adresse est de transformer un paquet en un autre paquet (possiblement identique). Chaque règle r de $\text{Post}_{\mathfrak{fw}}$ peut donc être vue comme une relation partielle $\overset{r}{\longmapsto}_{\text{Post}}$ de $\mathcal{T}(\Sigma)_{\mathsf{Packet}}$ dans $\mathcal{T}(\Sigma)_{\mathsf{Packet}}$.

Un raisonnement similaire au cas des règles de filtrage nous conduit à définir pour chaque règle $r \in \text{Post}_{\mathfrak{fw}}$ les relations $r \ll_{\text{Post}} p$ et $r \ll^{\mathfrak{fw}}_{\text{Post}} p$ signifiant respectivement que :

- p filtre les conditions de la règle r
- p filtre les conditions de la règle r et ne répond aux conditions d'aucune règle plus prioritaire $r' <_{\mathfrak{fw}} r$.

Proposition 5.9. Pour toute règle de post-routage r, $\overset{r}{\longmapsto}_{\text{Post}}$ est effectivement régulière.

Pour montrer cette proposition, nous avons besoin d'une propriété de fermeture particulière sur les relations régulières.

Lemme 5.10. Soit R une relation régulière et C un contexte. La relation $\{(C[t_1], \ldots, C[t_n]) \mid (t_1, \ldots, t_n) \in R\}$ est régulière (mais non fortement régulière).

❝ *Démonstration.* Notons $\mathbb{A} = (\Sigma^n, Q, F, \Delta)$ l'automate reconnaissant R. Soit $\{q_\omega \mid \omega \in \mathcal{P}os(C)\}$ un ensemble d'états disjoint de Q. Pour toute position ω de $\mathcal{P}os(C)$ telle que $C(\omega) \neq \Box$, on définit l'ensemble de règles $rec_\omega(C)$ suivant :

$$\langle\, C(\omega), \ldots, C(\omega) \,\rangle\, (q_1, \ldots, q_m) \to q_\omega$$

où m est l'arité du symbole $C(\omega)$, et pour tout $1 \leq i \leq m$, q_i est soit $q_{\omega.i}$ si $C(\omega.i) \neq \Box$, soit un élément de F sinon. L'automate

$$\mathbb{A}' = \left(\Sigma^n, Q \cup \{q_\omega \mid \omega \in \mathcal{P}os(C),\ C(\omega) \neq \Box\}, \{q_\varepsilon\}, \Delta \cup \bigcup_{\omega \in \mathcal{P}os(C), C(\omega) \neq \Box} rec_\omega(C) \right)$$

reconnaît la relation $\{(C[t_1], \ldots, C[t_n]) \mid (t_1, \ldots, t_n) \in R\}$. ❞

La démonstration précédente peut être adaptée en considérant des contextes comportant plusieurs « trous » (à remplir avec plusieurs relations régulières). On obtient alors la proposition suivante :

Proposition 5.11. Soit $R_1, \ldots, R_m$ des relations n-aires régulières et t un terme linéaire contenant exactement m variables $\{x_1, \ldots, x_m\}$ (vu comme un contexte à m trous). La relation

$$t[R_1, \ldots, R_m] = \{(\sigma_1(t), \ldots, \sigma_n(t)) \mid \forall i \in [1, m], (\sigma_1(x_i), \ldots, \sigma_n(x_i)) \in R_i\}$$

est régulière.

Démontrons maintenant la proposition 5.9.

❝ *Démonstration.* Soit $r = lhs \twoheadrightarrow rhs$ une règle de traduction d'adresse. Quatre cas sont possibles :

(i) lhs et rhs sont deux termes clos de sorte IP.
Dans ce cas, soient $t = \mathbf{packet}(x_1, x_2, x_3, x_4)$ un terme linéaire, Id_{s} la relation binaire régulière $\{(t, t) \mid t \in \mathcal{T}(\Sigma)_{\mathsf{s}}\}$ et $rec(lhs, rhs)$ la relation régulière $\{(lhs, rhs)\}$. La relation $\overset{r}{\longmapsto}_{\text{Post}}$ est alors égale à :

$$t[Id_{\mathsf{IP}}, Id_{\mathsf{Port}}, rec(lhs, rhs), Id_{\mathsf{port}}]$$

(*ii*) lhs et rhs sont deux termes clos de sorte Port, dans ce cas, la relation $\overset{r}{\longmapsto}_{\mathrm{Post}}$ est égale à :

$$t[Id_{\mathsf{IP}}, Id_{\mathsf{Port}}, Id_{\mathsf{IP}}, rec(lhs, rhs)]$$

(*iii*) lhs et rhs sont deux couples de termes clos (lhs_{ip}, lhs_{port}) et (rhs_{ip}, rhs_{port}) de sorte $\mathsf{IP} \times \mathsf{Port}$. Dans ce cas, la relation $\overset{r}{\longmapsto}_{\mathrm{Post}}$ est égale à :

$$t[Id_{\mathsf{IP}}, Id_{\mathsf{Port}}, rec(lhs_{ip}, rhs_{ip}), rec(lhs_{port}, rhs_{port})]$$

(*iv*) lhs est un terme de sorte IP contenant une variable et rhs est un terme clos de sorte IP. Nous avons vu que de telles règles expriment l'association entre l'ensemble de toutes les destinations dont l'adresse IP est caractérisée par le terme clos rhs et dont le port peut être n'importe quel port dont l'écriture binaire commence par 11 avec l'ensemble des adresses dénotées par lhs. Notons $rec(lhs)$ l'ensemble régulier des instances closes de lhs. $rec(lhs) \times \{rhs\}$ est un ensemble fortement régulier donc régulier. L'ensemble des ports dynamiques correspond à l'ensemble régulier $rec(\mathbf{port}(\mathbf{11}(x)))$ des instances closes de $\mathbf{port}(\mathbf{11}(x))$. La relation $\overset{r}{\longmapsto}_{\mathrm{Post}}$ peut donc s'exprimer de la façon suivante :

$$t[rec(lhs) \times \{rhs\}, rec(\mathbf{port}(y)) \times rec(\mathbf{port}(\mathbf{11}(x))), Id_{\mathsf{IP}}, Id_{\mathsf{Port}}]$$

❞

Comme dans le cas des règles de filtrage, la sémantique d'une règle de traduction d'adresse dépend de l'ensemble des autres règles de traduction plus prioritaires. Notons $\overset{r}{\longmapsto}^{\mathfrak{fw}}_{\mathrm{Post}}$ la sémantique d'une règle r compte-tenu des autres règles présentes dans $\mathrm{Pre}_{\mathfrak{fw}}$. Il s'agit d'une relation de $\mathcal{T}(\Sigma)_{\mathsf{Packet}}$ dans $\mathcal{T}(\Sigma)_{\mathsf{Packet}}$.

Proposition 5.12. Pour tout firewall $\mathfrak{fw}$ et pour toute règle $r \in \mathrm{Post}_{\mathfrak{fw}}$, la relation $\overset{r}{\longmapsto}^{\mathfrak{fw}}_{\mathrm{Post}}$ est effectivement régulière.

Lorsqu'aucune règle ne filtre un paquet donné, alors ce dernier n'est pas transformé. En d'autres termes, la règle s'appliquant par défaut est l'identité. On peut donc obtenir la sémantique de l'ensemble des règles SNAT d'un firewall à partir du calcul suivant :

$$\longmapsto^{\mathfrak{fw}}_{\mathrm{Post}} = \left(\bigcup_{r \in \mathrm{Post}_{\mathfrak{fw}}} \overset{r}{\longmapsto}^{\mathfrak{fw}}_{\mathrm{Post}} \right) \cup id_{|\mathsf{Packet} \setminus \left(\bigcup_{r \in \mathrm{Post}_{\mathfrak{fw}}} \ll_{\mathrm{Post}}(r) \right) \times \mathsf{Packet}}$$

On obtient ainsi le résultat suivant :

Proposition 5.13. Pour tout firewall $\mathfrak{fw}$, $\longmapsto^{\mathfrak{fw}}_{\mathrm{Post}}$ est effectivement régulier.

Toutes les définitions et propriétés énoncées pour SNAT sont valables pour DNAT. La sémantique des règles DNAT diffère cependant de celles de SNAT dans la mesure

où l'ensemble DNAT doit être augmenté d'une règle implicite effectuant la traduction inverse des règles de masquerading de DNAT. En notant $\text{Post}_{\mathfrak{fw}}^{masq}$ l'ensemble des règles de masquerading, la sémantique des règles DNAT est obtenue par le calcul suivant :

$$\begin{aligned}\longmapsto_{\text{Pre}}^{\mathfrak{fw}} = & \left(\bigcup_{r \in \text{Post}_{\mathfrak{fw}}^{masq}} \left(\overset{r}{\longmapsto}_{\text{Post}}^{\mathfrak{fw}} \right)^{-1} \right) \\ & \cup \left(\bigcup_{r \in \text{Pre}_{\mathfrak{fw}}} \overset{r}{\longmapsto}_{\text{Pre}}^{\mathfrak{fw}} \right)_{|\mathsf{Packet} \setminus \left(\bigcup_{r \in \text{Post}_{\mathfrak{fw}}^{masq}} \ll_{\text{Post}}(r) \right) \times \mathsf{Packet}} \\ & \cup id_{|\mathsf{Packet} \setminus \left(\bigcup_{r \in \text{Pre}_{\mathfrak{fw}}} \ll_{\text{Pre}}(r) \right) \times \mathsf{Packet}}\end{aligned}$$

On rappelle que si R est une relation binaire (fortement) régulière, alors R^{-1} l'est aussi.

Sémantique d'un firewall

Nous avons vu dans la section 5.1 que lorsqu'un paquet p passe à travers un firewall $\mathfrak{fw}$, les étapes suivantes sont appliquées successivement :

- p est réécrit en un autre paquet p' par les règles de préroutage (DNAT),
- p' est soit jeté soit accepté,
- si le paquet p' est accepté, alors il est réécrit en un autre paquet p'' par les règles de postroutage (SNAT).

On peut ainsi voir un firewall $\mathfrak{fw}$ comme une relation associant à tout paquet $p \in \mathcal{T}(\Sigma)_{\mathsf{Packet}}$ soit un autre paquet soit **drop**. Notons $\overset{\mathfrak{fw}}{\longmapsto}$ cette relation.

Proposition 5.14. Pour tout firewall $\mathfrak{fw}$, la relation $\overset{\mathfrak{fw}}{\longmapsto}$ est effectivement régulière.

❝ *Démonstration.*

$$\begin{aligned}\overset{\mathfrak{fw}}{\longmapsto} = & \sqcap_2 \big(\sqcup_3 (\longmapsto_{\text{Pre}}^{\mathfrak{fw}}) \cap \sqcup_{1,3} (\longmapsto_{\text{Filter}}^{\mathfrak{fw}})^{-1} (\{\textbf{accept})\} \cap \sqcup_1 (\longmapsto_{\text{Post}}^{\mathfrak{fw}}) \big) \\ & \cup \sqcap_2 \left(\longmapsto_{\text{Pre}}^{\mathfrak{fw}} \cap \sqcup_1 (\longmapsto_{\text{Filter}}^{\mathfrak{fw}})^{-1} (\{\textbf{drop})\} \right) \times \{\textbf{drop}\}\end{aligned}$$

❞

5.3 Applications

Nous allons montrer dans cette section que les analyses usuelles peuvent s'exprimer au moyen d'opérations (préservant la régularité) sur des ensembles et des relations dont nous avons montré la régularité. Nous montrerons également que notre approche nous permet d'envisager d'aller plus loin dans ces analyses.

5.3.1 Propriétés

Si l'on considère les firewalls comme des processus de décision associant à chaque paquet entrant une décision qui est soit **drop** soit un autre paquet, alors les propriétés suivantes sont pertinentes à analyser : la *cohérence*, indiquant qu'au plus une décision est prise pour un paquet donné, la *terminaison*, assurant qu'un firewall détermine les décisions en temps fini et la *complétude*, signifiant que pour tout paquet entrant, le firewall retourne une décision. Un autre problème qu'il est important de savoir traiter est la possibilité de comparer différents firewalls.

Par construction, chaque firewall caractérise un processus de décision qui termine (nombre fixe d'étapes de calcul) et est cohérent. Remarquons toutefois que la notion de cohérence mérite d'être précisée. En effet, les règles de masquerading permettent au firewall de traduire différemment un paquet entrant en fonction des ports dynamiques disponibles. Pour autant, nous considérons que les firewalls engendrent des processus de décision cohérents dans le sens où il n'est pas possible d'obtenir une décision positive et une décision négative pour un même paquet entrant. Bien que cohérents, les firewalls sont indéterministes lorsque des règles de masquerading sont présentes. Nous reviendrons sur les conséquences de l'indéterminisme lors de l'étude de la composition des firewalls. Le caractère déterministe des firewalls n'influe pas sur l'analyse d'un firewall indépendant. Ceci étant, focalisons nous sur la notion de complétude et sur la comparaison des firewalls. Pour cela, introduisons la notion de domaine de décision.

Définition 5.15. On appelle **domaine de décision** d'un firewall $\mathfrak{fw}$ l'ensemble des paquets auxquels $\mathfrak{fw}$ associe une décision, autrement dit $\mathcal{D}om(\overset{\mathfrak{fw}}{\longmapsto})$.

Proposition 5.16. Pour tout firewall $\mathfrak{fw}$, $\mathcal{D}om(\overset{\mathfrak{fw}}{\longmapsto})$ est un ensemble effectivement régulier.

Il s'agit de la projection sur la première composante de $\overset{\mathfrak{fw}}{\longmapsto} : \sqcap_1(\overset{\mathfrak{fw}}{\longmapsto})$.

Définition 5.17. Un firewall $\mathfrak{fw}$ est dit **complet** ssi $\mathcal{D}om(\overset{\mathfrak{fw}}{\longmapsto}) = \mathcal{T}(\Sigma)_{\mathsf{Packet}}$.

Dans la mesure où $\mathcal{D}om(\overset{\mathfrak{fw}}{\longmapsto})$ et $\mathcal{T}(\Sigma)_{\mathsf{Packet}}$ sont deux ensembles réguliers, on obtient la propriété suivante :

Proposition 5.18. La complétude est décidable.

Dans le cas de firewalls complets, il peut être important de déterminer si un firewall est plus ou moins permissif qu'un autre. Par « plus permissif » on entend « autorise au moins le même traffic ». Un tel ordre n'est évidemment pas total.

Définition 5.19. On définit l'**ordre partiel** $\preceq$ sur l'ensemble des firewalls complets de la façon suivante : pour tous firewalls complets $\mathfrak{fw}$ et $\mathfrak{fw}'$, $\mathfrak{fw} \preceq \mathfrak{fw}'$ ($\mathfrak{fw}'$ est plus permissif que $\mathfrak{fw}$) ssi pour tous $p, p' \in \mathcal{T}(\Sigma)_{\mathsf{Packet}}$, si $p \overset{\mathfrak{fw}}{\longmapsto} p'$ alors $p \overset{\mathfrak{fw}'}{\longmapsto} p'$. On écrira $\mathfrak{fw} \approx \mathfrak{fw}'$ ssi $\mathfrak{fw} \preceq \mathfrak{fw}'$ et $\mathfrak{fw}' \preceq \mathfrak{fw}$. Notez que $\approx$ est une relation d'équivalence et que $\mathfrak{fw} \approx \mathfrak{fw}'$ ssi $\overset{\mathfrak{fw}}{\longmapsto} = \overset{\mathfrak{fw}'}{\longmapsto}$.

Proposition 5.20. La relation d'ordre $\preceq$ (et en conséquence la relation d'équivalence $\approx$) est décidable.

❝ *Démonstration.* Il suffit de calculer les automates reconnaissant les restrictions de $\overset{\mathfrak{fw}}{\longmapsto}$ et $\overset{\mathfrak{fw}'}{\longmapsto}$ à l'ensemble des paquets qui ne sont pas associés à **drop** et de tester l'inclusion entre ces deux relations (vues comme des ensembles). ❞

5.3.2 Analyse structurelle

Le principe de l'analyse structurelle consiste à détecter les erreurs de configurations (aussi appelées anomalies) dans les règles d'un firewall. La notion d'erreur de configurations correspond à la présence de relations suspicieuses entre les règles d'un firewall. Si certaines de ces relations manifestent sans aucun doute des anomalies de conception (par exemple lorsqu'une règle filtre un ensemble de paquets déjà filtrés par d'autres règles plus prioritaires), certaines relations n'indiquent que de potentielles erreurs. C'est le cas notamment des règles filtrant un ensemble de paquets dont une partie seulement a déjà été filtrée par une règle plus prioritaire. Si la présence de telles règles peut dénoter une potentielle erreur, cela peut également traduire une simple préférence de l'administrateur dans sa façon de configurer les règles. Notre objectif n'est pas de discuter de la pertinence des anomalies traitées dans la littérature ni même d'en proposer de nouvelles. Nous souhaitons en revanche montrer que notre approche est particulièrement adaptée à l'étude de ces dernières. Pour un aperçu complet des anomalies, nous conseillons au lecteur de consulter les articles suivants : [Cuppens et al., 2006, Hamed and Al-Shaer, 2006]. La figure 5.1 rappelle l'ensemble des relations que nous avons montré régulières (les relations ternaires comportent un point $\cdot$ indiquant l'emplacement du second argument).

Il est bien connu que le calcul du premier ordre est intimement lié à l'algèbre relationnelle [Codd, 1972] et que l'ensemble des solutions d'une formule du premier ordre dans une interprétation donnée peut être calculé au moyen d'opérations de l'algèbre relationnelle sur les interprétations des relations. Par ailleurs, nous avons évoqué dans un précédent chapitre que les opérations fondamentales de l'algèbre relationnelle (opérations booléennes, projections, cylindrification, produit cartésien) préservent de façon effective la régularité des relations et que les relations et ensembles de bases (ensemble des termes d'une sorte donnée et relation identité) sont réguliers. Ceci nous permet d'envisager la définition de propriétés sous la forme de formules du premier ordre et, en considérant les firewalls comme des interprétations des symboles de prédicats utilisés, l'obtention automatique de l'algorithme de vérification correspondant. Nous définissons dans ce but la théorie suivante :

$$
\begin{array}{lll}
(i) \ll_{\text{Filter}} & (ii) \ll_{\text{Post}} & (iii) \ll_{\text{Pre}} \\
(iv) \ll^{\mathfrak{fw}}_{\text{Filter}} & (v) \ll^{\mathfrak{fw}}_{\text{Post}} & (vi) \ll^{\mathfrak{fw}}_{\text{Pre}} \\
(vii) \overset{\cdot}{\longmapsto}_{\text{Filter}} & (viii) \overset{\cdot}{\longmapsto}_{\text{Post}} & (ix) \overset{\cdot}{\longmapsto}_{\text{Pre}} \\
(x) \overset{\mathfrak{fw}}{\underset{\text{Filter}}{\longmapsto}} & (xi) \overset{\mathfrak{fw}}{\underset{\text{Post}}{\longmapsto}} & (xii) \overset{\mathfrak{fw}}{\underset{\text{Pre}}{\longmapsto}} \\
(xiii) \longmapsto^{\mathfrak{fw}}_{\text{Filter}} & (xiv) \longmapsto^{\mathfrak{fw}}_{\text{Post}} & (xv) \longmapsto^{\mathfrak{fw}}_{\text{Pre}} \\
(xvi) \overset{\mathfrak{fw}}{\longmapsto} & &
\end{array}
$$

FIGURE 5.1 – Relations associées à la théorie d'un firewall $\mathfrak{fw}$

Définition 5.21. Pour tout firewall $\mathfrak{fw}$, on définit $\mathscr{T}_{\mathfrak{fw}}$ comme étant la théorie du premier ordre basée sur la syntaxe suivante :

- Constantes : règles de $\mathfrak{fw}$, termes clos de sorte Packet, décisions (**accept** et **drop**)
- Symboles de prédicats : symboles donnés par la figure 5.1 et $<_{\mathfrak{fw}}$
- Quantificateurs : $\exists$ et $\forall$
- Variables : tout ensemble dénombrable
- Connecteurs : $\wedge, \vee, \Rightarrow, \neg$

et dont la sémantique est la suivante :

- les termes sont interprétés trivialement par eux-mêmes,
- les prédicats sont interprétés par les relations présentées dans la section précédente.

Les ensembles et relations de la théorie sont soit finis (ensembles des règles, $<_{\mathfrak{fw}}$), soit réguliers. En conséquence, on obtient la proposition suivante :

Proposition 5.22. Pour tout firewall $\mathfrak{fw}$, la théorie $\mathscr{T}_{\mathfrak{fw}}$ est décidable (ce qui signifie que l'on peut déterminer la validité de toute formule de la théorie). De plus, pour toute formule de $\mathscr{T}_{\mathfrak{fw}}$, on peut calculer un automate d'arbres reconnaissant l'ensemble des solutions de la formule (*i.e.* l'ensemble des valuations des variables libres rendant la formule vraie).

Un algorithme possible pour calculer l'automate associé à une formule de $\mathscr{T}_{\mathfrak{fw}}$ peut être établi en suivant l'approche décrite dans [Dauchet and Tison, 1990]. En substance, il s'agit d'effectuer les opérations suivantes :

- mettre la formule sous forme polie ;
- linéariser les atomes
  $(p(\ldots, x, \ldots, x, \ldots)$ étant équivalent à $\exists y, p(\ldots, x, \ldots, y, \ldots) \wedge x = y)$;
- sortir les termes clos des atomes
  $(p(\ldots, t, \ldots)$ étant équivalent à $\exists y, p(\ldots, y, \ldots) \wedge y = t)$;
- transformer tous les quantificateurs universels en quantificateurs existentiels ;
- mettre la formule obtenue en forme normale disjonctive (*i.e.* sous la forme de disjonctions de conjonctions) ;

- descendre les quantificateurs existentiels le plus bas possible dans l'arbre de la formule ;
- remplacer toute égalité entre variables $x = y$ par l'atome $Id(x, y)$ (Id étant interprété par la relation régulière identité) ;
- remplacer toute égalité entre une variable et un terme clos $x = t$ par l'atome $rec_t(x)$ où rec_t est interprété par l'ensemble (régulier) $\{t\}$;
- remplacer les négations par l'opérateur de complémentarité ($\neg p(\ldots)$ devient $\overline{p}(\ldots)$) ;
- cylindrifier les atomes en conjonction (ou disjonction) de telle sorte à ce qu'ils aient exactement les mêmes variables ($p(x, \ldots) \wedge q(y, \ldots)$ devient $(\sqcup_2 p)(x, y, \ldots) \wedge (\sqcup_1 q)(x, y, \ldots)$) ;
- intervertir les composantes des atomes en conjonction (ou disjonction) de telle sorte à ce que leurs variables apparaissent dans le même ordre (l'opération d'inversion des composantes préserve la régularité) ;
- remplacer les conjonctions par des intersections ($p(x_1, \ldots, x_n) \wedge q(x_1, \ldots, x_n)$ devient $(p \cap q)(x_1, \ldots, x_n)$) ;
- remplacer les quantificateurs existentiels par des projections ($\exists x_k, p(x_1, \ldots, x_n)$ devient $(\sqcap_k p)(x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n)$) ;
- remplacer les disjonctions par des unions.

L'application de ces règles jusqu'à l'obtention d'un seul atome permet d'obtenir le calcul à effectuer sur les automates correspondant aux interprétations des prédicats pour obtenir l'automate reconnaissant les valeurs des variables libres validant la formule.

> *Exemple 5.23.* Soient p un prédicat binaire interprété par une relation binaire régulière, t un terme clos et φ la formule $p(y, t) \wedge \forall z.(p(z, t) \Rightarrow (z = y \vee p(z, y)))$. L'application des règles précédentes transforme cette formule en l'expression suivante :
>
> $$\underbrace{\left(\sqcap_2(p \cap \sqcup_1 rec_t) \cap \overline{\sqcap_1 \left(\left(\sqcup_2(\sqcap_2(p \cap \sqcup_1 rec_t)) \cap \overline{Id} \cap \overline{p}\right)\right)}\right)}_{\vartheta}(y)$$
>
> L'évaluation de l'expression ϑ retourne un automate d'arbres reconnaissant l'ensemble des valeurs t telles que $\{y \mapsto t\}(\varphi)$ soit vrai (dans l'interprétation de Herbrand dans laquelle p est interprété par un automate d'arbres du même nom).

Par ailleurs, nous pouvons aisément montrer que les anomalies présentées dans la littérature peuvent s'exprimer comme des formules de $\mathscr{T}_{\mathfrak{fw}}$. Il serait bien entendu fastidieux et peu pertinent de procéder à cet exercice pour toutes les anomalies usuelles, c'est pourquoi nous nous contenterons de traiter un exemple d'erreur de configuration particulier : le shadowing. Rappelons en premier lieu sa définition :

Définition 5.24. Un firewall possède la propriété de **shadowing** (ou de masquage) ssi il contient une règle de filtrage telle que tous les paquets acceptés (resp. refusés) par cette règle sont refusés (resp. acceptés) par une règle plus prioritaire. Dans ce

cas, la règle concernée est dite masquée.

Proposition 5.25. La propriété de masquage peut être exprimée par une formule de $\mathscr{T}_{\mathfrak{fw}}$.

En effet, si l'on note $Shadowed_{\mathfrak{fw}}(r)$ la propriété indiquant que la règle r est masquée, alors $Shadowed_{\mathfrak{fw}}(r)$ peut être défini par la formule suivante :

$$\begin{aligned} & \forall p\colon p \xmapsto[\text{Filter}]{r}{}^{\mathfrak{fw}}\mathbf{accept} \Rightarrow (\exists r'\colon r' <_{\mathfrak{fw}} r \wedge p \xmapsto[\text{Filter}]{r'}{}^{\mathfrak{fw}}\mathbf{drop}) \\ \vee \quad & \forall p\colon p \xmapsto[\text{Filter}]{r}{}^{\mathfrak{fw}}\mathbf{drop} \Rightarrow (\exists r'\colon r' <_{\mathfrak{fw}} r \wedge p \xmapsto[\text{Filter}]{r'}{}^{\mathfrak{fw}}\mathbf{accept}) \end{aligned}$$

Dans ce cadre, la question qui se pose ensuite est de savoir si la démarche proposée est raisonnablement efficace. Il est en effet connu que les opérations sur les automates d'arbres sont d'une complexité élevée en général. On peut cependant remarquer que dans notre cas la complexité des opérations est principalement liée à la représentation des adresses. Focalisons donc notre discussion sur les adresses et leur représentation. Nous avons fait le choix de décrire les adresses comme des mots $\{0, 1\}$ (ou de façon équivalente des termes construits sur les symboles monadiques **0** et **1** et sur la constante #). Pour simplifier les explications, nous considerons les automates de mots (la correspondance avec les automates d'arbres s'obtient, comme évoqué dans la section précédente, en inversant l'ordre de lecture des symboles). Une bonne propriété des ensembles d'adresses manipulés est que l'automate déterministe minimal correspondant possède une forme particulière : il ne comporte aucune boucle à l'exception de son unique état final bouclant sur tous les symboles de l'alphabet. Qualifions un langage de n-préfixe (ou simplement préfixe) lorsqu'il est de la forme $\alpha_1.A^* \cup \ldots \cup \alpha_n.A^*$ où A est un alphabet et α_i sont des mots sur A. On utilise ici la notation usuelle des expressions régulières de mots. A^* représente l'ensemble des mots de taille quelconque sur A. Notons $\mathcal{P}ref_A^n$ la classe des langages n-préfixes sur l'alphabet A et $\mathcal{P}ref_A = \bigcup_{n\geq 0} \mathcal{P}ref_A^n$.

Proposition 5.26. La classe $\mathcal{P}ref_A$ est stable pour les opérations booléennes.

De plus, ces opérations peuvent être effectuées en complexité linéaire.

Proposition 5.27. Soient L_1 et L_2 deux langages préfixes. Alors, l'automate minimal déterministe reconnaissant $L_1 \oplus L_2$ (resp. $\overline{L_1}$), où $\oplus$ est une opération booléenne, peut être calculé en $O(n_1 + n_2)$ (resp. $O(n_1)$) où n_i est le nombre de transitions de l'automate minimal déterministe reconnaissant L_i.

❝ *Démonstration.* Nous effectuons la démonstration en utilisant les notations relatives aux automates d'arbres. Soit $\mathbb{A} = (\Sigma, Q_{\mathbb{A}}, F_{\mathbb{A}}, \Delta_{\mathbb{A}})$ et $\mathbb{B} = (\Sigma, Q_{\mathbb{B}}, F_{\mathbb{B}}, \Delta_{\mathbb{B}})$ deux automates d'arbres finis minimaux déterministes reconnaissant deux langages préfixes sur les symboles **0** et **1** et la constante #. La construction de $\mathbb{C} = \mathbb{A} \cap \mathbb{B} = (\Sigma, Q_{\mathbb{C}}, F_{\mathbb{C}}, \Delta_{\mathbb{C}})$

s'obtient par l'application des règles suivantes (c-à-d la construction des plus petits ensembles $Q_{\mathbb{C}}, F_{\mathbb{C}}$ et $\Delta_{\mathbb{C}}$ tels que le dénominateur de chaque règle est vérifié pour toute valeur rendant vrai le numérateur correspondant) :

$$(i) \quad \frac{\# \rightarrow_{\mathbb{A}} q \quad ; \quad \# \rightarrow_{\mathbb{B}} q'}{(q,q') \in Q_{\mathbb{C}} \quad ; \quad \# \rightarrow_{\mathbb{C}} (q,q')}$$

$$(ii) \quad \frac{(q,q') \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a} \in \{\mathbf{0},\mathbf{1}\} \quad ; \quad \mathbf{a}(q) \rightarrow_{\mathbb{A}} p \quad ; \quad \mathbf{a}(q') \rightarrow_{\mathbb{B}} p'}{(p,p') \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a}(q,q') \rightarrow_{\mathbb{C}} (p,p')}$$

$$(iii) \quad \frac{(q,q') \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a} \in \{\mathbf{0},\mathbf{1}\} \quad ; \quad \mathbf{a}(q) \rightarrow_{\mathbb{A}} p \quad ; \quad q' \in F_{\mathbb{B}}}{(p,q') \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a}(q,q') \rightarrow_{\mathbb{C}} (p,q')}$$

$$(iv) \quad \frac{(q,q') \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a} \in \{\mathbf{0},\mathbf{1}\} \quad ; \quad q \in F_{\mathbb{A}} \quad ; \quad \mathbf{a}(q') \rightarrow_{\mathbb{B}} p'}{(q,p') \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a}(q,p') \rightarrow_{\mathbb{C}} (q,p')}$$

$$(v) \quad \frac{(q,q') \in Q_{\mathbb{C}} \quad ; \quad q \in F_{\mathbb{A}} \quad ; \quad q' \in F_{\mathbb{B}}}{(q,q') \in F_{\mathbb{C}} \quad ; \quad \mathbf{0}(q,q') \rightarrow_{\mathbb{C}} (q,q') \quad ; \quad \mathbf{1}(q,q') \rightarrow_{\mathbb{C}} (q,q')}$$

La construction de $\mathbb{C} = \mathbb{A} \cup \mathbb{B}$ s'obtient par l'application des règles suivantes :

$$(i) \quad \frac{\# \rightarrow_{\mathbb{A}} q \quad ; \quad \# \rightarrow_{\mathbb{B}} q'}{(q,q') \in Q_{\mathbb{C}} \quad ; \quad \# \rightarrow_{\mathbb{C}} (q,q')}$$

$$(ii) \quad \frac{(q,q') \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a} \in \{\mathbf{0},\mathbf{1}\} \quad ; \quad \mathbf{a}(q) \rightarrow_{\mathbb{A}} p \quad ; \quad \mathbf{a}(q') \rightarrow_{\mathbb{B}} p'}{(p,p') \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a}(q,q') \rightarrow_{\mathbb{C}} (p,p')}$$

$$(iii) \quad \frac{(q,q') \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a} \in \{\mathbf{0},\mathbf{1}\} \quad ; \quad \mathbf{a}(q) \rightarrow_{\mathbb{A}} p \quad ; \quad \nexists p', \mathbf{a}(q') \rightarrow_{\mathbb{B}} p'}{(p,p_{fresh}) \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a}(q,q') \rightarrow_{\mathbb{C}} (p,p_{fresh})}$$

$$(iv) \quad \frac{(q,q') \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a} \in \{\mathbf{0},\mathbf{1}\} \quad ; \quad \nexists p, \mathbf{a}(q) \rightarrow_{\mathbb{A}} p \quad ; \quad \mathbf{a}(q') \rightarrow_{\mathbb{B}} p'}{(p_{fresh},p') \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a}(q,q') \rightarrow_{\mathbb{C}} (p_{fresh},p)}$$

$$(v) \quad \frac{(q,q') \in Q_{\mathbb{C}} \quad ; \quad q \in F_{\mathbb{A}}}{(q,q') \in F_{\mathbb{C}} \quad ; \quad \mathbf{0}(q,q') \rightarrow_{\mathbb{C}} (q,q') \quad ; \quad \mathbf{1}(q,q') \rightarrow_{\mathbb{C}} (q,q')}$$

$$(v) \quad \frac{(q,q') \in Q_{\mathbb{C}} \quad ; \quad q' \in F_{\mathbb{B}}}{(q,q') \in F_{\mathbb{C}} \quad ; \quad \mathbf{0}(q,q') \rightarrow_{\mathbb{C}} (q,q') \quad ; \quad \mathbf{1}(q,q') \rightarrow_{\mathbb{C}} (q,q')}$$

Dans les deux cas, il s'agit d'effectuer un parcours « parallèle » des deux automates $\mathbb{A}$ et $\mathbb{B}$ et d'appliquer la règle qui correspond à la description des états courants.

Enfin, la construction de $\mathbb{C} = \overline{\mathbb{A}}$ s'obtient par l'application des règles suivantes :

$$(i) \qquad \frac{\# \rightarrow_{\mathbb{A}} q \quad ; \quad q \notin F_{\mathbb{A}}}{q \in Q_{\mathbb{C}} \quad ; \quad \# \rightarrow_{\mathbb{C}} q}$$

$$(ii) \qquad \frac{q \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a} \in \{\mathbf{0}, \mathbf{1}\} \quad ; \quad \mathbf{a}(q) \rightarrow_{\mathbb{A}} q' \quad ; \quad q' \notin F_{\mathbb{A}}}{q' \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a}(q) \rightarrow_{\mathbb{C}} q'}$$

$$(iii) \qquad \frac{q \in Q_{\mathbb{C}} \quad ; \quad \mathbf{a} \in \{\mathbf{0}, \mathbf{1}\} \quad ; \quad \nexists q', \mathbf{a}(q) \rightarrow_{\mathbb{A}} q'}{q_{fresh} \in Q_{\mathbb{C}} \text{ et } F_{\mathbb{C}} \quad ; \quad \mathbf{0}(q_{fresh}) \rightarrow_{\mathbb{C}} q_{fresh} \quad ; \quad \mathbf{1}(q_{fresh}) \rightarrow_{\mathbb{C}} q_{fresh}}$$

Les automates obtenus sont déterministes mais pas minimaux. Pour les rendres minimaux, il suffit d'effectuer un parcours inversé de l'automate obtenu en fusionnant les transitions « similaires » : il s'agit d'une sorte de déterminisation de l'automate inversé, c'est à dire dans lequel les états finaux sont considérés comme initiaux mais qui ne nécessite que $n \leq |\, Q_{\mathbb{C}} \,|$ pas puisque chaque transition est impliquée dans exactement une étape de déterminisation. Il suffit ensuite d'appliquer la règle suivante :

$$\frac{\mathbf{0}(q) \rightarrow q' \quad ; \quad \mathbf{1}(q) \rightarrow q' \quad ; \quad F = \{q'\}}{F_{min} = \{q\} \quad ; \quad q' \notin Q_{min}}$$

qui ne nécessite qu'un seul parcours des transitions de l'automate considéré. ❞

5.3.3 Analyse par requêtage

Le second principal type d'analyse traité dans la littérature est l'analyse par requêtage. Nous avons déjà évoqué ce type d'analyse dans les chapitres précédents. Comme précédemment, ce type d'analyse a pour objectif d'assister l'administrateur d'un firewall dans sa maîtrise du comportement de ce dernier en lui permettant d'évaluer des requêtes de son choix du type « quels hôtes du sous-réseau 192.168.1.1/22 peuvent recevoir des paquets de la part d'un hôte du sous-réseau 172.20.1.1/24 ? ». De nombreux travaux se sont attachés à élaborer des algorithmes dédiés à l'analyse par requêtage dans le cas des firewalls [Marmorstein and Kearns, 2005], [Liu and Gouda, 2009]. Pour capturer la classe de requêtes présentée par exemple dans [Liu and Gouda, 2009], il est suffisant d'étendre légèrement l'ensemble des formules de la théorie $\mathscr{T}_{\mathfrak{fw}}$. En l'état, $\mathscr{T}_{\mathfrak{fw}}$ ne permet pas de raisonner sur les différents champs des paquets (adresses source et destination, ...). Nous ajoutons donc les éléments syntaxiques suivants à $\mathscr{T}_{\mathfrak{fw}}$:

- les termes linéaires de sorte Packet dont les variables sont de sorte IP ou Port,
- les sous-réseaux (vus comme des constantes),
- un prédicat binaire $\in$ entre les termes de sorte Packet et les sous-réseaux.

Cette extension de $\mathscr{T}_{\mathfrak{fw}}$ préserve la proposition 5.22.

> *Exemple 5.28.* Considérons un firewall $\mathfrak{fw}$ séparant un réseau privé d'Internet. Nous souhaitons déterminer à partir de quelle adresse IP du réseau interne et de quel port il est possible d'accéder à la page web de l'INRIA. Étant entendu que l'adresse IP du site de l'INRIA est 193.51.193.149 et que celle du réseau

interne est 192.168.1.1/22, notre requête peut s'exprimer de la façon suivante :

$$\varphi(ip_1, p_1) = \begin{cases} ip_1 \in 192.168.1.1/22 \wedge \exists ip_2, p_2: \\ \quad packet(ip_1, p_1, ip_2, 80) \overset{\text{fw}}{\longmapsto} packet(ip_2, p_2, 193.51.193.149, 80) \end{cases}$$

5.4 Composition de firewalls

Une politique de sécurité réseau est généralement déployée en utilisant plusieurs firewalls. Nous avons étudié dans la section précédente les méthodes d'analyse de firewalls lorsqu'ils sont considérés isolemment. La composition de firewalls va généralement au delà de la simple juxtaposition et fait intervenir des mécanismes complexes comme le routage. Les conséquences de l'application de ces mécanismes sont par ailleurs dépendantes du fonctionnement des règles de traduction d'adresses (ce qui explique que l'on parle de pré- et de post-routage). On comprend alors qu'il n'est en général pas suffisant d'analyser des firewalls dans leur individualité pour s'assurer que leur composition satisfasse les propriétés exigées. Bien que le routage puisse engendrer des problèmes de sécurité majeurs, il s'agit d'un élément souvent négligé dans les processus d'analyse de politiques de sécurité réseau. Ils peuvent par exemple engendrer des boucles dans le processus d'acheminement des paquets et provoquer en conséquence des problèmes de congestion voire de déni de service. De plus, l'interaction entre les règles de routage et les règles des firewalls peut aboutir à un acheminement différent des paquets en fonction du chemin qu'ils empruntent, rendant ainsi la politique de sécurité ambigüe.

Les effets induits par l'interaction entre le processus de routage et les firewalls rend la sémantique de la politique de sécurité globale difficile à appréhender et ce, particulièrement dans les réseaux importants dont la topologie est complexe. Nous nous proposons dans cette section de spécifier la topologie du réseau ainsi que les règles de routage afin de pouvoir analyser la politique de sécurité globale du réseau.

5.4.1 Topologies de réseaux

Dans l'optique de spécifier la topologie du réseau ainsi que la composition des firewalls, il est nécessaire de modéliser de nouveaux éléments : l'identification des sous-réseaux, la localisation des hôtes de sécurité (c'est à dire les noeuds du réseau dans lesquels sont déployés les firewalls en tant qu'entités logiques) ainsi que les connections entre les sous-réseaux et les hôtes de sécurité.

Nous l'avons déjà évoqué, un sous-réseau peut être vu comme une unité logique consistant en un ensemble d'hôtes qui peuvent communiquer mutuellement sans passer par l'intermédiaire d'un hôte de sécurité. Il correspond généralement à une section particulière d'une organisation et est usuellement identifié par un nom symbolique ainsi qu'un intervalle d'adresses (son domaine). Les hôtes de sécurité sont des noeuds d'interconnection du réseau qui peuvent être connectés à des sous-réseaux (auquel cas l'hôte de sécurité sera appelée la passerelle [5] du sous-réseau en question) ou à

5. Notons que sans perte de généralité, on peut considérer que tout sous-réseau possède exactement une passerelle. Dans le cas où nous souhaiterions modéliser une topologie ne vérifiant pas cette propriété, il serait suffisant d'ajouter un hôte de sécurité intermédiaire.

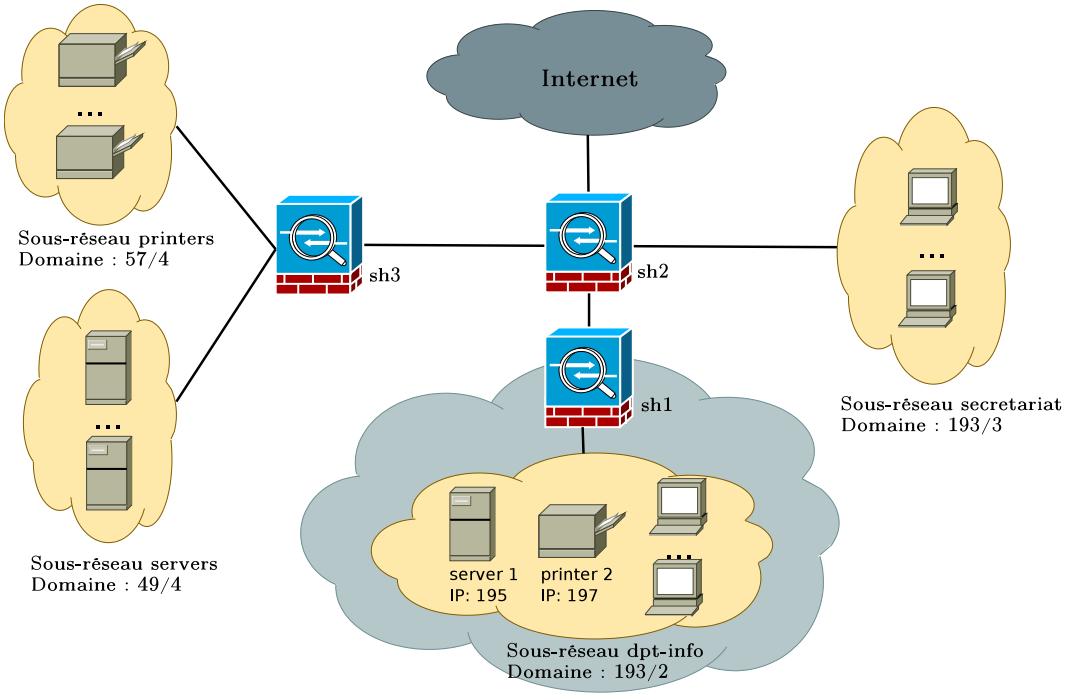


FIGURE 5.2 – Exemple de topologie réseau

d'autres hôtes de sécurité.

La figure 5.2 donne un exemple de topologie de réseau.

Pour des raisons de lisibilité, nous considérons dans cette figure ainsi que dans les exemples qui lui sont associés que les adresses IP sont construits sur un seul octet (c'est à dire ne comporte que 8 bits) et que le domaine d'Internet est 129/2.

Dans l'optique de spécifier formellement une telle topologie, nous ajoutons à la signature Σ la sorte Net pour représenter les noms de sous-réseaux ainsi que la sorte SH pour représenter les noms des hôtes de sécurité.

Définition 5.29. Une **topologie réseau** τ est donnée par une extension de la signature Σ, notée Σ_τ comprenant :

- un ensemble fini de constantes $\mathbf{net_1}, \ldots, \mathbf{net_n} : \rightarrow \mathsf{Net}$ de noms de sous-réseau ;
- un ensemble fini de constantes $\mathbf{s_1}, \ldots, \mathbf{s_m} : \rightarrow \mathsf{SH}$ de noms d'hôtes de sécurité ;

ainsi que

- une application $\Delta : \mathcal{T}(\Sigma_\tau)_{\mathsf{Net}} \rightarrow \mathcal{T}(\Sigma_\tau, \mathcal{X})_{\mathsf{IP}}$ associant à chaque nom de sous-réseau $\mathbf{net_i}$ un terme linéaire de $\mathcal{T}(\Sigma_\tau, \mathcal{X})_{\mathsf{IP}}$ (son domaine) ;
- une application $\mathcal{GW} : \mathcal{T}(\Sigma_\tau)_{\mathsf{Net}} \rightarrow \mathcal{T}(\Sigma_\tau)_{\mathsf{SH}}$ associant à tout nom de sous-réseau $\mathbf{net_i}$ un hôte de sécurité (sa passerelle) telle que pour tout hôte de sécurité $\mathbf{s}$ et tous sous-réseaux distincts $\mathbf{net_1}, \mathbf{net_2} \in \mathcal{GW}^{-1}(\mathbf{s})$, les termes $\Delta(\mathbf{net_1})$ et $\Delta(\mathbf{net_2})$ ne sont pas unifiables (autrement dit, les intervalles

d'adresses de différents sous-réseaux connectés au même hôte de sécurité sont nécessairement disjoints) ;
- une relation irréflexive, transitive et symétrique $\mathcal{CON}$ sur $\mathcal{T}(\Sigma_\tau)_{\mathsf{SH}} \times \mathcal{T}(\Sigma_\tau)_{\mathsf{SH}}$ décrivant la connection entre les hôtes de sécurité.

Exemple 5.30. La topologie décrite dans la figure 5.2 est formellement définie de la façon suivante :
- $\mathbf{printers}, \mathbf{servers}, \mathbf{secretariat}, \mathbf{dpt{-}info} : \rightarrow \mathsf{Net}$;
- $\mathbf{sh_1}, \mathbf{sh_2}, \mathbf{sh_3} : \rightarrow \mathsf{SH}$;
- $\Delta = \left\{ \begin{array}{rcl} \mathbf{printers} & \mapsto & \mathbf{ip}(57/4[x]) \\ \mathbf{servers} & \mapsto & \mathbf{ip}(49/4[x]) \\ \mathbf{secretariat} & \mapsto & \mathbf{ip}(193/3[x]) \\ \mathbf{dpt{-}info} & \mapsto & \mathbf{ip}(193/2[x]) \end{array} \right\}$;
- $\mathcal{GW} = \left\{ \begin{array}{rcl} \mathbf{printers} & \mapsto & \mathbf{sh_3} \\ \mathbf{servers} & \mapsto & \mathbf{sh_3} \\ \mathbf{secretariat} & \mapsto & \mathbf{sh_2} \\ \mathbf{dpt{-}info} & \mapsto & \mathbf{sh_1} \end{array} \right\}$;
- $\mathcal{CON} = \{(\mathbf{sh_1}, \mathbf{sh_2}), (\mathbf{sh_2}, \mathbf{sh_1}), (\mathbf{sh_2}, \mathbf{sh_3}), (\mathbf{sh_3}, \mathbf{sh_2})\}$.

5.4.2 Politiques de sécurité réseau

Étant donnée une topologie réseau, définir une politique de sécurité réseau consiste à définir pour chaque hôte de sécurité un ensemble de règles de filtrages et de traduction d'adresses (autrement dit un firewall). Les firewalls associés aux hôtes de sécurité décrivent un ensemble de politiques de sécurité « locales » qui sont combinées par la topologie. Par ailleurs, nous considérions implicitement, dans les sections précédentes, que les firewalls séparaient exactement deux sous-réseaux. Dans la suite, nous ajoutons aux firewalls la notion d'interface permettant à un firewall de séparer un nombre arbitraire de réseaux. (Notez bien que si cela simplifie la suite de la présentation, cela n'augmente pas le pouvoir d'expression : un firewall avec n interfaces pouvant être simulé avec $\frac{n \times (n-1)}{2}$ firewalls « binaires »). La définition d'un firewall est alors raffinée comme suit :

Définition 5.31. Un **firewall** $\mathfrak{fw}$ est la donnée :
- d'un ensemble ordonné (ordre noté $<_{\mathfrak{fw}}$) $\mathrm{Filter}_{\mathfrak{fw}}$ de règles vérifiant les mêmes conditions que celles décrites dans la définition 5.2 ;
- d'un ensemble fini $\mathrm{If}_{\mathfrak{fw}}$ dont les éléments sont appelés interfaces ;
- pour toute interface $if \in \mathrm{If}_{\mathfrak{fw}}$, de deux ensembles ordonnés (ordre également noté $<_{\mathfrak{fw}}$) $\mathrm{Pre}_{\mathfrak{fw}}(if)$ et $\mathrm{Post}_{\mathfrak{fw}}(if)$ de règles vérifiant les mêmes conditions que celles décrites dans la définition 5.2.

Une politique de sécurité réseau s'obtient donc en associant à chaque hôte de sécurité un firewall dont chaque interface correspond à une connection de l'hôte de sécurité.

Définition 5.32. Une **politique de sécurité réseau** $\mathfrak{pol}$ sur une topologie τ est une application associant à tout hôte de sécurité **sh** de τ un firewall $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$ ainsi qu'une bijection $\mu^{\mathbf{sh}}_{\mathfrak{fw}}$ entre $\mathcal{CON}(\mathbf{sh}) \cup \mathcal{GW}^{-1}(\mathbf{sh})$ et $\mathrm{If}_{\mathfrak{fw}}$ ($\mu^{\mathbf{sh}}_{\mathfrak{fw}}(\mathbf{sh}')$ est l'interface de $\mathfrak{fw}$ associée à la connection de **sh** à **sh**′).

Exemple 5.33. Soit τ la topologie définie dans l'exemple 5.30. Nous définissons la politique de sécurité $\mathfrak{pol}$ sur τ de la façon suivante :

- l'objectif de l'hôte de sécurité $\mathbf{sh_1}$ est de cacher l'espace des adresses IP privée du sous-réseau **dpt−info** ainsi que les hôtes de ce sous-réseaux à l'exception de l'imprimante identifiée « printer 2 » et du serveur « server 1 ». Pour atteindre cet objectif, $\mathbf{sh_1}$ est associé au firewall $\mathfrak{fw}_1 = \mathfrak{pol}(\mathbf{sh_1})$ constitué des règles suivantes :
  - $\mathrm{Pre}_{\mathfrak{fw}_1}(\mu^{\mathbf{sh_1}}_{\mathfrak{fw}_1}(\mathbf{sh_2})) = \left\{ \begin{array}{rcl} \mathbf{ip}(19), \mathbf{port}(515) & \twoheadrightarrow & \mathbf{ip}(197), \mathbf{port}(515) \\ \mathbf{ip}(17), \mathbf{port}(8080) & \twoheadrightarrow & \mathbf{ip}(195), \mathbf{port}(32) \end{array} \right.$
  - $\mathrm{Post}_{\mathfrak{fw}_1}(\mu^{\mathbf{sh_1}}_{\mathfrak{fw}_1}(\mathbf{sh_2})) = \mathrm{Pre}_{\mathfrak{fw}_1}(\mu^{\mathbf{sh_1}}_{\mathfrak{fw}_1}(\mathbf{sh_2}))^{-1} \cup \{\mathbf{ip}(193/2[x]) \twoheadrightarrow \mathbf{ip}(18)\}$
  - $\mathrm{Post}_{\mathfrak{fw}_1}(\mu^{\mathbf{sh_1}}_{\mathfrak{fw}_1}(\mathbf{dpt{-}info})) = \mathrm{Pre}_{\mathfrak{fw}_1}(\mu^{\mathbf{sh_1}}_{\mathfrak{fw}_1}(\mathbf{dpt{-}info})) = \varnothing$
  - $\mathrm{Filter}_{\mathfrak{fw}_1} = \left\{ \begin{array}{rcl} \mathbf{packet}(\mathbf{ip}(197), \mathbf{port}(515), x, y) & \twoheadrightarrow & \mathbf{accept} \\ \mathbf{packet}(\mathbf{ip}(195), \mathbf{port}(32), x, y) & \twoheadrightarrow & \mathbf{accept} \\ \mathbf{packet}(x, y, \mathbf{ip}(197), \mathbf{ip}(515)) & \twoheadrightarrow & \mathbf{accept} \\ \mathbf{packet}(x, y, \mathbf{ip}(195), \mathbf{port}(32)) & \twoheadrightarrow & \mathbf{accept} \\ \mathbf{packet}(\mathbf{ip}(197), x, y, z) & \twoheadrightarrow & \mathbf{drop} \\ \mathbf{packet}(\mathbf{ip}(195), x, y, z) & \twoheadrightarrow & \mathbf{drop} \\ \mathbf{packet}(x, y, \mathbf{ip}(197), z) & \twoheadrightarrow & \mathbf{drop} \\ \mathbf{packet}(x, y, \mathbf{ip}(195), z) & \twoheadrightarrow & \mathbf{drop} \\ \mathbf{packet}(\mathbf{ip}(193/2[x]), 80, y, z) & \twoheadrightarrow & \mathbf{accept} \\ \mathbf{packet}(x, y, z, u) & \twoheadrightarrow & \mathbf{drop} \end{array} \right.$

  L'imprimante du département informatique (printer 2) est accessible sous l'adresse 19 et uniquement sur le port *515* (port associé au protocole d'impression Line Printer Daemon) tandis que le serveur de ce département (server 1) est accessible sous l'adresse IP *17* sur le port *32* mais visible sous le port *8080*. Les autres hôtes de ce sous-réseau peuvent communiquer avec le protocole HTTP (port 80).
- $\mathbf{sh_2}$ est associé au firewall $\mathfrak{pol}(\mathbf{sh_2})$ contenant les règles de filtrage suivantes :

  $$\left\{ \begin{array}{rrcl} (i) & \mathbf{packet}(\mathbf{ip}(193/3[x]), x', \mathbf{ip}(129/2[y]), y') & \rightarrow & \mathbf{drop} \\ (ii) & \mathbf{packet}(\mathbf{ip}(129/2[x]), x', y, z) & \rightarrow & \mathbf{drop} \\ (iii) & \mathbf{packet}(x, y, z, u) & \rightarrow & \mathbf{accept} \end{array} \right.$$

  décrivant le fait que : (i) les hôtes du sous-réseau secrétariat ne peuvent pas initier de communication internet ; (ii) aucune communication ne peut être initiée d'Internet à destination d'un hôte du réseau interne ; (iii) tous les autres paquets peuvent être transmis.
- Le troisième hôte de sécurité $\mathbf{sh_3}$ se focalise sur le trafic envoyé ou reçu par les sous-réseaux **printers** et **servers**. Il est associé au firewall

$\mathfrak{pol}(\mathbf{sh_3})$ contenant les règles de filtrage suivantes :

$$\begin{cases} (i) & \mathbf{packet}(\mathbf{ip}(193/3[x]), x', \mathbf{ip}(57/4[x]), x') & \twoheadrightarrow & \mathbf{accept} \\ (ii) & \mathbf{packet}(\mathbf{ip}(97/3[x]), x', y, z) & \twoheadrightarrow & \mathbf{accept} \\ (iii) & \mathbf{packet}(\mathbf{ip}(17/4[x]), x', y', z) & \twoheadrightarrow & \mathbf{accept} \\ (iv) & \mathbf{packet}(x, x', z, u) & \twoheadrightarrow & \mathbf{drop} \end{cases}$$

signifiant que : (i) les hôtes du sous-réseau **secretariat** peuvent atteindre le sous-réseaux des imprimantes mais pas celui des serveurs ; (ii) l'hôte de sécurité ne bloque pas le traffic émis des imprimantes et des serveurs ($97/3 = 57/4 \cup 49/4$) ; (iii) les paquets émis de **dpt−info** (vus par $\mathbf{sh_3}$ comme le sous-réseau de domaine $17/4$) est autorisé et (iv) tout autre trafic est interdit.

5.4.3 Stratégie de routage

Pour entièrement caractériser le trafic réseau régi par une politique de sécurité donnée, il est nécessaire de munir la politique d'une stratégie de routage pour déterminer les chemins que les paquets doivent emprunter pour atteindre leur destination. Il s'agit donc d'une stratégie de composition des firewalls.

Définition 5.34. Étant donnée une topologie réseau τ, une **stratégie de routage** est une application $\zeta : \mathcal{T}(\Sigma_\tau)_{\mathsf{SH}} \to \mathcal{T}(\Sigma_\tau, \mathcal{X})_{\mathsf{IP}} \to \mathcal{T}(\Sigma_\tau)_{\mathsf{SH}}$ telle que :

i) pour tous $(\mathbf{sh}, t_1, \mathbf{sh'_1})$ et $(\mathbf{sh}, t_2, \mathbf{sh'_2}) \in \zeta$, il n'existe aucune substitution σ telle que $\sigma(t_1) = \sigma(t_2)$ et

ii) pour tout $(\mathbf{sh}, t, \mathbf{sh'}) \in \zeta$, il n'existe aucun $\mathbf{net} \in \mathcal{GW}^{-1}(\mathbf{sh})$ ni substitution σ tels que $\sigma(\Delta(\mathbf{net})) = \sigma(t)$.

Une stratégie de routage ζ associe à tout hôte de sécurité une table de routage indiquant le prochain hôte de sécurité auquel transmettre un paquet en fonction de sa destination. Pour modéliser raisonnablement les processus de routage présents dans la réalité, nous imposons que (i) le routage soit déterministe, c.-à-d. tous les hôtes de sécurité associent à tout paquet entrant au plus un autre hôte de sécurité et (ii) que les paquets dont la destination appartient à un réseau directement accessible doit être délivré (et ne doit donc pas être routé vers un autre hôte de sécurité). Par ailleurs, l'objectif d'une stratégie de routage consiste à transmettre tout paquet d'un hôte (de sécurité) à un autre hôte (de sécurité) jusqu'à ce qu'il atteigne un hôte capable de prendre une décision finale, c'est-à-dire délivrer le message ou le jeter. Conformément à cette vision, une stratégie de routage doit satisfaire certaines conditions. Tout d'abord, elle ne doit pas créer de boucle lors de la transmission d'un paquet. Ensuite, elle doit nécessairement transmettre tout paquet vers un hôte pouvant prendre une décision (si aucune décision ne peut être prise, le paquet est perdu). Ainsi, nous dirons dans la suite qu'une stratégie de routage ζ est *correcte* relativement à une topologie τ si elle satisfait les conditions précédemment mentionnées. Formellement, ζ est correcte relativement à $\tau = (\mathsf{Net}, \mathsf{SH}, \Delta, \mathcal{GW}, \mathcal{CON})$ ssi pour tout $t \in \bigcup_{\mathbf{net} \in \mathcal{T}(\Sigma_\tau)_{\mathsf{Net}}} rec(\Delta(\mathbf{net}))$ et $\mathbf{sh} \in \mathcal{T}(\Sigma_\tau)_{\mathsf{SH}}$:

- il existe une séquence finie $(\mathbf{sh_i}, t_i, \mathbf{sh_{i+1}})_{i=1\ldots n}$ de tuples de ζ ainsi qu'une séquence de substitutions $(\sigma_i)_{i=1\ldots n+1}$ telle que $\mathbf{sh_1} = \mathbf{sh}$ et pour tout $1 \leq i \leq n : (\mathbf{sh_i}, \mathbf{sh_{i+1}}) \in \mathcal{CON}$, $t = \sigma_i(t_i)$ et $t = \sigma_{n+1}(\Delta(\mathbf{net}))$ pour un certain **net** tel que $\mathcal{GW}(\mathbf{net}) = \mathbf{sh_{n+1}}$ et
- il n'existe pas de séquences $(\mathbf{sh_i}, t_i, \mathbf{sh_{i+1}})_{i=1\ldots n}$ et $(\sigma_i)_{i=1\ldots n}$ telles que $\mathbf{sh_1} = \mathbf{sh_{n+1}} = \mathbf{sh}$ et pour tout $1 \leq i \leq n : (\mathbf{sh_i}, \mathbf{sh_{i+1}}) \in \mathcal{CON}$ and $t = \sigma_i(t_i)$.

Dans ce qui suit, nous considérerons que toutes les stratégies de routage considérées sont correctes relativement aux topologies par rapport auxquelles elles sont définies.

Exemple 5.35. Suivant le précédent exemple, nous définissons la stratégie de routage ci-dessous :

- $\zeta(\mathbf{sh_1}) = \left\{ \begin{array}{rcl} \mathbf{ip}(125/1[x]) & \mapsto & \mathbf{sh_2} \\ \mathbf{ip}(32/3[x]) & \mapsto & \mathbf{sh_2} \end{array} \right.$; $\mathbf{ip}(64/2[x]) \mapsto \mathbf{sh_2}$; $\mathbf{ip}(1/4[x]) \mapsto \mathbf{sh_2}$ et
  $\zeta(\mathbf{sh_3}) = \left\{ \begin{array}{rcl} \mathbf{ip}(128/1[x]) & \mapsto & \mathbf{sh_2} \\ \mathbf{ip}(64/3[x]) & \mapsto & \mathbf{sh_2} \end{array} \right.$; $\mathbf{ip}(1/2[x]) \mapsto \mathbf{sh_2}$
  spécifiant que tous les paquets envoyés à une adresse qui n'est pas directement atteignable de $\mathbf{sh_1}$ (resp. $\mathbf{sh_3}$) est redirigée vers $\mathbf{sh_2}$ et
- $\zeta(\mathbf{sh_2}) = \{\ 97/3[x] \mapsto \mathbf{sh_3}\ ;\ 17/4[x] \mapsto \mathbf{sh_1}$

5.4.4 Sémantique par réécriture

Les questions d'intérêt sont les mêmes que celles qui ont été soulevées lors de l'étude des firewalls seuls. Quels paquets atteignent leur destination finale ? Lesquel sont jetés au cours de leur acheminement ? Cependant, l'analyse de ce type de questions est plus complexe dans le cas présent. En effet, par opposition au cas des firewalls simples où la décision d'accepter ou de jeter un paquet est évaluée localement, dans le cas d'une combinaison de firewalls dans une topologie réseau, il existe potentiellement un certain nombre d'étapes intermédiaires entre le moment où un paquet est envoyé et celui où il est reçu par son destinataire (ou jeté).

En d'autres termes, un paquet peut revêtir différents états : émis, reçu, sur le point d'être filtré par un hôte de sécurité, ... La figure 5.3 représente un exemple d'états successifs dans lesquels peut se trouver un paquet p.

Pour analyser le trafic sous une politique de sécurité donnée, notre approche consiste à étiqueter les paquets pour indiquer l'état courant dans lequel ils se trouvent et de décrire l'évolution de leur état sous la forme d'un processus de réécriture. Pour représenter les états, nous introduisons les nouveaux symboles suivants :

$$
\begin{array}{lll}
\mathbf{sent} & : & \mathsf{Net} \times \mathsf{IP} \times \mathsf{Port} \times \mathsf{IP} \times \mathsf{Port} \rightarrow \mathsf{State} \\
\mathbf{received} & : & \mathsf{Net} \times \mathsf{IP} \times \mathsf{Port} \times \mathsf{IP} \times \mathsf{Port} \rightarrow \mathsf{State} \\
\mathbf{dropped} & : & \rightarrow \mathsf{State} \\
\mathbf{pre} & : & \mathsf{SH} \cup \mathsf{Net} \times \mathsf{SH} \times \mathsf{IP} \times \mathsf{Port} \times \mathsf{IP} \times \mathsf{Port} \rightarrow \mathsf{State} \\
\mathbf{post} & : & \mathsf{SH} \times \mathsf{SH} \cup \mathsf{Net} \times \mathsf{IP} \times \mathsf{Port} \times \mathsf{IP} \times \mathsf{Port} \rightarrow \mathsf{State} \\
\mathbf{forward} & : & \mathsf{SH} \times \mathsf{SH} \cup \mathsf{Net} \times \mathsf{IP} \times \mathsf{Port} \times \mathsf{IP} \times \mathsf{Port} \rightarrow \mathsf{State} \\
\mathbf{filter} & : & \mathsf{SH} \times \mathsf{IP} \times \mathsf{Port} \times \mathsf{IP} \times \mathsf{Port} \rightarrow \mathsf{State} \\
\mathbf{route} & : & \mathsf{SH} \times \mathsf{IP} \times \mathsf{Port} \times \mathsf{IP} \times \mathsf{Port} \rightarrow \mathsf{State}
\end{array}
$$

où $\mathsf{SH} \cup \mathsf{Net}$ est vue comme une sorte telle que $\mathsf{SH} \leq \mathsf{SH} \cup \mathsf{Net}$ et $\mathsf{Net} \leq \mathsf{SH} \cup \mathsf{Net}$ indiquant que tout terme de sorte SH ou de sorte Net est de sorte $\mathsf{SH} \cup \mathsf{Net}$. Pour être

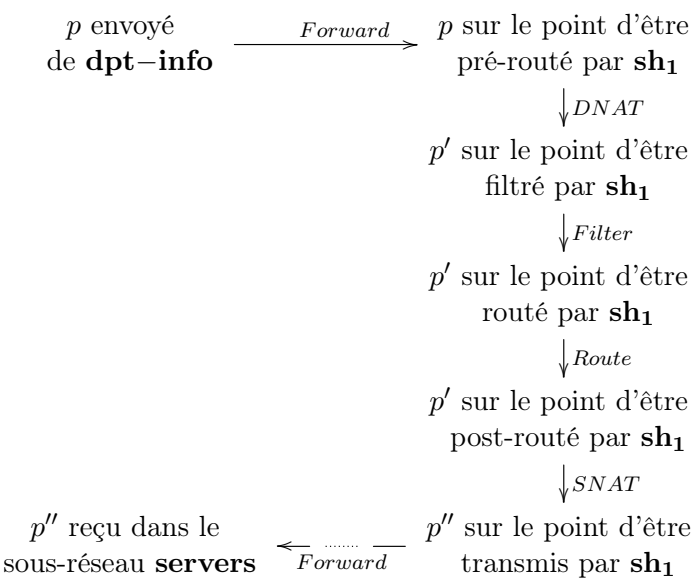


FIGURE 5.3 – Exemple du processus d'acheminement d'un paquet p

conforme à l'idée classique selon laquelle une politique de sécurité évalue une requête d'accès en une décision, nous utilisons, pour toute topologie réseau, le vocabulaire suivant :

- une requête d'accès réseau fait référence à un terme clos de la forme

$$\mathbf{sent}(\mathbf{net}, t_1, t_2, t_3, t_4)$$

  où $\mathbf{net} \in \mathcal{T}(\Sigma_\tau)_{\mathsf{Net}}$ et où les termes clos t_1, t_2, t_3 et t_4 sont tels que $t_1 = \sigma(t)$ pour $t = \Delta(\mathbf{net})$ et pour une certaine substitution close σ ;
- une décision fait référence à **dropped** ou à un terme clos dont le symbole de tête est **received**.

Nous noterons $\mathcal{R}equest$ l'ensemble des requêtes d'accès réseau et $\mathcal{D}ecision$ l'ensemble des décisions. (En toute rigueur, la désignation de ces ensembles devrait être paramétrée par la topologie τ considérée mais dans la suite la détermination de cette dernière sera non ambigüe.) Dans ce contexte, chaque politique de sécurité réseau $\mathfrak{pol}$ induit une relation associant à chaque requête zéro, une ou plusieurs décisions. Pour pouvoir analyser cette relation, nous nous proposons de la décrire au moyen d'un système de réécriture possédant des caractéristiques particulières. Commençons par introduire la classe de systèmes de réécriture que nous allons considérer.

Définition 5.36. Une règle de réécriture $l \to r$ est dite **croissante** si elle est linéaire et si lorsque $l(\omega) = r(\omega') \in \mathcal{X}$ pour deux positions ω, ω', alors $|\,\omega\,| \leq 1$. Un **système de réécriture croissant** est un ensemble de règles de réécriture croissantes.

Définition 5.37. Un **système de réécriture contraint à contraintes fortement régulières** (SRC-RS) est un système de réécriture dont chaque règle $l \to r$ est associée à un ensemble de contraintes d'appartenances $x \in A$ où x est une variable de $\mathcal{V}ar(l)$ et A est un ensemble régulier. Lorsque $l \to r$ est associé à $\{x_1 \in A_1, \ldots, x_n \in A_n\}$, on note $l \to r \ \parallel \ x_1 \in A_1, \ldots, x_n \in A_n$.

La relation de réduction $\to_{\mathrm{R}}$ engendrée par un SRC-RS R est définie de la façon suivante : pour tous termes t, t', $t \to_{\mathrm{R}} t'$ ssi il existe une règle $l \to r \ \parallel \ x_1 \in A_1, \ldots, x_n \in A_n$ de R, un contexte C ainsi qu'une substitution σ tels que $t = C[\sigma(l)]$, $t' = C[\sigma(r)]$ et $\sigma(x_i) \in A_i$ pour tout i. Notez que de façon strictement équivalente, on peut considérer des contraintes du type $\#(x_1, \ldots, x_n) \in A$ où A est une relation fortement régulière (une règle de cette forme est équivalente à un ensemble fini de règles de la forme présentée dans la définition 5.37).

Nous nous intéressons dans la suite à la classe des systèmes de réécriture contraints croissants à contraintes fortement régulières que nous noterons SRC-GRS. Nous nous proposons de modéliser la sémantique d'une politique de sécurité réseau sous la forme d'un tel système de réécriture afin d'en permettre l'analyse. Comme nous l'avons déjà évoqué, chaque étape de l'évaluation d'une requête d'accès (filtrage, traduction d'adresse, routage, ...) transforme un paquet étiqueté en un autre paquet étiqueté jusqu'à l'obtention d'une forme normale (paquet reçu ou jeté). Chacune de ses étapes peut alors être vue comme une relation de $\mathcal{T}(\Sigma_\tau)_{\mathsf{State}}$ dans $\mathcal{T}(\Sigma_\tau)_{\mathsf{State}}$. On suppose données une topologie τ, une politique de sécurité réseau $\mathfrak{pol}$ sur τ ainsi qu'une stratégie de routage ζ. La première étape du processus d'évaluation d'une requête d'accès est l'émission d'un paquet $\mathbf{packet}(t_1, t_2, t_3, t_4)$ à partir d'un sous-réseau **net**. Lorsque le paquet est émis, il est modélisé sous la forme du terme $\mathbf{sent}(\mathbf{net}, t_1, t_2, t_3, t_4)$. Dès lors, deux cas sont possibles. Soit la destination appartient au même sous-réseau que celui duquel le paquet a été émis et dans ce cas le processus d'émission transforme $\mathbf{sent}(\mathbf{net}, t_1, t_2, t_3, t_4)$ en $\mathbf{received}(\mathbf{net}, t_1, t_2, t_3, t_4)$, indiquant que le paquet a été reçu, soit la destination n'appartient pas à **net** et alors le paquet doit être transmis à la passerelle du sous-réseau d'émission. Dans ce cas, $\mathbf{sent}(\mathbf{net}, t_1, t_2, t_3, t_4)$ devient $\mathbf{pre}(\mathbf{net}, \mathbf{sh}, t_1, t_2, t_3, t_4)$, indiquant que le paquet provient de **net** et doit procéder à l'étape de pré-routing de l'hôte de sécurité **sh**. Cette étape peut être spécifiée par le schéma de règles de réécriture (contraintes croissantes à contraintes fortement régulières) suivant :

Broadcast

(B) $\mathbf{sent}(\mathbf{net}, x, y, z, u) \ \to \ \mathbf{received}(\mathbf{net}, x, y, z, u) \ \parallel \ z \in rec(\Delta(net))$
pour tout $\mathbf{net} \in \mathsf{Net}$

Gateway

(G) $\mathbf{sent}(\mathbf{net}, x, y, z, u) \ \to \ \mathbf{pre}(\mathbf{net}, \mathbf{sh}, x, y, z, u) \ \parallel \ z \in \overline{rec(\Delta(net))}$
pour tout $\mathbf{net} \in \mathsf{Net}$ *et* $\mathbf{sh} = \mathcal{GW}(\mathbf{net})$

Le processus de pré-routage, ou DNAT, transforme les paquets de la forme $\mathbf{pre}(\mathbf{net}, \mathbf{sh}, t_1, t_2, t_3, t_4)$ ou $\mathbf{pre}(\mathbf{sh}', \mathbf{sh}, t_1, t_2, t_3, t_4)$ en un terme dont le symbole de tête est **filter** indiquant que le paquet doit être filtré par l'hôte **sh**. Deux cas sont

possibles : soit il existe une règle de traduction et alors l'état obtenu après traduction est un terme de la forme $\mathbf{filter}(\mathbf{sh}, t_1, t_2, t'_3, t'_4)$ avec $t'_3 \neq t_3$ ou $t'_4 \neq t_4$ (t_3 et/ou t_4 ont été transformés), soit il n'existe aucune règle de traduction applicable au paquet et alors $\mathbf{pre}(\mathbf{net}, \mathbf{sh}, t_1, t_2, t_3, t_4)$ devient $\mathbf{filter}(\mathbf{sh}, t_1, t_2, t_3, t_4)$. Dans le premier cas, pour obtenir un SRC-GRS, il est nécessaire de modéliser différemment le processus de traduction en fonction de la règle DNAT appliquée. Pour cela, il est nécessaire d'introduire quelques ensembles et relations (fortement réguliers) auxiliaires. Notons $\mathcal{D}om^{\mathfrak{fw}}_{\mathrm{NAT}}(r)$ l'ensemble régulier des couples $(ip, port)$ impactés par la règle de traduction r. Dans le cas où $r \in \mathrm{Post}_{\mathfrak{fw}}$, cet ensemble est défini par :

$$\mathcal{D}om_{\mathrm{NAT}}(r) \setminus \bigcup_{r' <_{\mathfrak{fw}} r \in \mathrm{Post}_{\mathfrak{fw}}} \mathcal{D}om_{\mathrm{NAT}}(r')$$

et, si $r \in \mathrm{Pre}_{\mathfrak{fw}}$, par :

$$\mathcal{D}om_{\mathrm{NAT}}(r) \setminus \left(\bigcup_{r' <_{\mathfrak{fw}} r \in \mathrm{Pre}_{\mathfrak{fw}}} \mathcal{D}om_{\mathrm{NAT}}(r') \cup \overbrace{\bigcup_{ip[x] \twoheadrightarrow ip' \in \mathrm{Post}_{\mathfrak{fw}}} rec(ip') \times rec(\mathbf{11}(x))}^{\mathcal{D}om^{\mathfrak{fw}}_{masq}} \right)$$

où

$$\mathcal{D}om_{\mathrm{NAT}}(r) = \begin{cases} \{ip\} \times \mathcal{T}(\Sigma)_{\mathsf{Port}} & si\ r = ip \twoheadrightarrow ip' \\ \mathcal{T}(\Sigma)_{\mathsf{IP}} \times \{port\} & si\ r = port \twoheadrightarrow port' \\ \{ip\} \times \{port\} & si\ r = ip, port \twoheadrightarrow ip', port' \\ rec(ip[x]) \times \mathcal{T}(\Sigma)_{\mathsf{Port}} & si\ r = ip[x] \twoheadrightarrow ip' \end{cases}$$

$\mathcal{D}om^{\mathfrak{fw}}_{masq}$ représentant l'ensemble des paquets impactés par les règles implicites correspondant aux règles de masquerading du post-routage.

Le processus de pré-routage peut être modélisé de la façon suivante :

Destination NAT

(D_1) $\mathbf{pre}(\mathbf{sn}, \mathbf{sh}, x, y, z, u) \rightarrow \mathbf{filter}(\mathbf{sh}, x, y, ip', u) \quad \| \quad (z, u) \in \mathcal{D}om^{\mathfrak{fw}}_{\mathrm{NAT}}(r)$
pour tout $\mathbf{sn} \in \mathsf{Net} \cup \mathsf{SH}$, $\mathbf{sh} \in \mathsf{SH}$, $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$
$r = ip \twoheadrightarrow ip' \in \mathrm{Pre}_{\mathfrak{fw}}(\mu^{\mathbf{sh}}_{\mathfrak{fw}}(\mathbf{sn}))$

(D_2) $\mathbf{pre}(\mathbf{sn}, \mathbf{sh}, x, y, z, u) \rightarrow \mathbf{filter}(\mathbf{sh}, x, y, z, port') \quad \| \quad (z, u) \in \mathcal{D}om^{\mathfrak{fw}}_{\mathrm{NAT}}(r)$
pour tout $\mathbf{sn} \in \mathsf{Net} \cup \mathsf{SH}$, $\mathbf{sh} \in \mathsf{SH}$, $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$
$r = port \twoheadrightarrow port' \in \mathrm{Pre}_{\mathfrak{fw}}(\mu^{\mathbf{sh}}_{\mathfrak{fw}}(\mathbf{sn}))$

(D_3) $\mathbf{pre}(\mathbf{sn}, \mathbf{sh}, x, y, z, u) \rightarrow \mathbf{filter}(\mathbf{sh}, x, y, ip', port') \quad \| \quad (z, u) \in \mathcal{D}om^{\mathfrak{fw}}_{\mathrm{NAT}}(r)$
pour tout $\mathbf{sn} \in \mathsf{Net} \cup \mathsf{SH}$, $\mathbf{sh} \in \mathsf{SH}$, $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$
$r = ip, port \twoheadrightarrow ip', port' \in \mathrm{Pre}_{\mathfrak{fw}}(\mu^{\mathbf{sh}}_{\mathfrak{fw}}(\mathbf{sn}))$

(D_4) $\mathbf{pre}(\mathbf{sn}, \mathbf{sh}, x, y, z, u) \rightarrow \mathbf{filter}(\mathbf{sh}, x, y, z, u)$
$\| \quad (z, u) \in \overline{\bigcup_{r \in \mathrm{Pre}_{\mathfrak{fw}}} \mathcal{D}om^{\mathfrak{fw}}_{\mathrm{NAT}}(r) \cup \mathcal{D}om^{\mathfrak{fw}}_{masq}}$
pour tout $\mathbf{sn} \in \mathsf{Net} \cup \mathsf{SH}$, $\mathbf{sh} \in \mathsf{SH}$, $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$

Après la traduction des adresses de destination, le paquet doit subir l'étape de filtrage. Cette étape se modélise naturellement par les règles suivantes :

Filter

(F_1) $\mathbf{filter}(\mathbf{sh}, x, y, z, u) \rightarrow \mathbf{dropped}$
$\quad \| \quad \mathbf{packet}(x, y, z, u) \in (\longmapsto^{\mathfrak{fw}}_{\mathrm{Filter}})^{-1}(\mathbf{dropped})$
pour tout $\mathbf{sh} \in \mathsf{SH}$ *où* $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$

(F_2) $\mathbf{filter}(\mathbf{sh}, x, y, z, u) \rightarrow \mathbf{route}(\mathbf{sh}, x, y, z, u)$
$\quad \| \quad \mathbf{packet}(x, y, z, u) \in (\longmapsto^{\mathfrak{fw}}_{\mathrm{Filter}})^{-1}(\mathbf{accept})$
pour tout $\mathbf{sh} \in \mathsf{SH}$ *où* $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$

La relation $\longmapsto^{\mathfrak{fw}}_{\mathrm{Filter}}$ étant fortement régulière, il est en de même pour $(\longmapsto^{\mathfrak{fw}}_{\mathrm{Filter}})^{-1}$ et en conséquence, les contraintes $\mathbf{packet}(x, y, z, u) \in (\longmapsto^{\mathfrak{fw}}_{\mathrm{Filter}})^{-1}(\mathbf{dropped})$ et $\mathbf{packet}(x, y, z, u) \in (\longmapsto^{\mathfrak{fw}}_{\mathrm{Filter}})^{-1}(\mathbf{accept})$ sont équivalentes à un ensemble fini de conjonctions de contraintes de la forme $x \in A_1, y \in A_2, z \in A_3, u \in A_4$. À la suite du filtrage l'hôte de sécurité doit déterminer le prochain hôte auquel transmettre le paquet. Il s'agit de l'étape de routage.

Route

(R_1) $\mathbf{route}(\mathbf{sh}, x, y, z, u) \rightarrow \mathbf{post}(\mathbf{sh}, \mathbf{sh}', x, y, z, u) \quad \| \quad z \in rec(t)$
pour tout $(\mathbf{sh}, t, \mathbf{sh}') \in \zeta$

(R_2) $\mathbf{route}(\mathbf{sh}, x, y, z, u) \rightarrow \mathbf{post}(\mathbf{sh}, \mathbf{net}, x, y, z, u) \quad \| \quad z \in rec(\Delta(net))$
pour tout $\mathbf{net} \in \mathsf{Net}$ *et* $\mathbf{sh} = \mathcal{GW}(\mathbf{net})$

Dans le premier cas, le routage indique le prochain hôte à qui transmettre le paquet en fonction de la stratégie de routage. La seconde règle modélise le cas où l'hôte de sécurité courant est la passerelle du réseau destinataire du paquet. Après la détermination du prochain hôte, le paquet doit subir l'étape de post-routage (SNAT) correspondant à l'interface du firewall connectée à l'hôte en question. Si dans le cas des règles de traduction de type one-to-one, la modélisation du processus de traduction sous la forme d'un système de réécriture SRC-GRS ne pose pas de problème particulier, la modélisation des règles de masquerading soulèvent quant à elles quelques difficultés. Comme nous l'avons déjà évoqué, un paquet traduit par ce type de règle peut être transformé en une infinité de paquets (l'adresse IP étant fixe mais le port de traduction étant indéterminé). Si l'on effectuait une modélisation sous la forme d'un système de réécriture, la variable correspondant au port de destination dans le membre droit des règles n'apparaîtrait pas en partie gauche (le port n'étant pas déterminé par le membre gauche). Pour pallier ce problème, il est nécessaire de fixer le port de traduction et de vérifier que cela n'affecte pas le comportement de la politique globale. De façon plus générale, il est nécessaire que tous les paquets, issus d'une étape de masquerading et pouvant représenter un même paquet initial, soient traités exactement de la même façon par les firewalls suivants. Par exemple, si un firewall contient la règle de masquerading suivante : $192.168.1.1/25 \twoheadrightarrow 172.1.1.1$,

un même paquet en provenance de 192.168.1.1 : 80 pourra être traduit en un paquet dont la source est 172.1.1.1 : 49152, 172.1.1.1 : 49153,..., ou 172.1.1.1 : 65535. Pour assurer la cohérence de la politique globale, c'est-à-dire le fait qu'une même requête d'accès aboutisse toujours à la même décision, il est nécessaire (mais pas suffisant), que les firewalls suivants traitent exactement de la même façon tous ces paquets modélisant potentiellement le même paquet initial. Nous appellerons cette condition « propriété de cohérence faible ». Pour définir formellement cette propriété, il est nécessaire d'introduire quelques notions auxiliaires. La première est la notion de stabilité d'une classe d'équivalence par rapport à une relation.

Définition 5.38. Soit E une relation d'équivalence sur un ensemble Ω et R une relation binaire sur $\Omega \times \Omega$. On dit que E est **stable** par R ssi :

$$\forall (x, y) \in E,\ R(\{x\}) = R(\{y\})$$

Cette propriété indique que tous les éléments équivalents dans E ont la même image par R.

Lorqu'un même paquet peut être transmis sous plusieurs formes différentes par une interface donnée d'un firewall, alors ces différentes formes forment une classe d'équivalence pour le trafic émis à partir de cette interface. La question que l'on se pose est de savoir si cette classe d'équivalence est stable pour les relations induites par les firewalls suivants. Révisons la définition de cette relation ($\overset{\mathfrak{fw}}{\longmapsto}$) présentée pour l'analyse des firewalls simples pour tenir compte de la notion d'interface : on note $\overset{\mathfrak{fw}}{\longmapsto}_{if,if'}$ la relation induite par la traversée d'un paquet de l'interface if du firewall $\mathfrak{fw}$ à son interface if' (c'est-à-dire la relation définie dans la section 5.2.2 en considérant $\mathrm{Pre}_{\mathfrak{fw}}(if)$ et $\mathrm{Post}_{\mathfrak{fw}}(if')$). Sont équivalents à la sortie d'une interface if d'un firewall $\mathfrak{fw}$ les paquets qui sont l'image d'un même paquet initial, autrement dit, les paquets $p_i \in \mathcal{T}(\Sigma)_{\mathsf{Packet}}$ tels qu'il existe une interface if' de $\mathfrak{fw}$ et un paquet $p \in \mathcal{T}(\Sigma)_{\mathsf{Packet}}$ tels que $\forall i, p \overset{\mathfrak{fw}}{\longmapsto}_{if',if} p_i$. Plus précisément, on établit la définition suivante :

Définition 5.39. Soit $\mathfrak{pol}$ une politique sur une topologie τ. Pour tout firewall $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$ et toutes interfaces if, if' de $\mathfrak{fw}$, on définit les relations d'**équivalence** suivantes :

(*i*) $\forall p, p' \in \mathcal{T}(\Sigma)_{\mathsf{Packet}}, p \approx_{if'/if} p'$ ssi il existe $q, q' \in \mathcal{T}(\Sigma)_{\mathsf{Packet}}$ tels que $q \approx^{in}_{if} q'$ et $q \overset{\mathfrak{fw}}{\longmapsto}_{if,if'} p$ et $q' \overset{\mathfrak{fw}}{\longmapsto}_{if,if'} p'$;

(*ii*) $\forall p, p' \in \mathcal{T}(\Sigma)_{\mathsf{Packet}}, p \approx^{out}_{if'} p'$ ssi il existe une interface if telle que $p \approx_{if'/if} p'$.

(*iii*) s'il existe $\mathbf{net} \in \mathsf{Net}$ tel que $\mu^{\mathbf{sh}}_{\mathfrak{fw}}(\mathbf{net}) = if$, alors $\approx^{in}_{if}$ est l'identité, c'est-à-dire $Id = \{(t, t) \in \mathcal{T}(\Sigma)^2_{\mathsf{Packet}}\}$.

(*iv*) s'il existe $\mathbf{sh}' \in \mathsf{SH}$ tel que $\mu^{\mathbf{sh}}_{\mathfrak{fw}}(\mathbf{sh}') = if$, et $\mu^{\mathbf{sh}'}_{\mathfrak{fw}}(\mathbf{sh}) = if'$, alors $p \approx^{in}_{if} p'$ ssi $p \approx^{out}_{if'} p'$.

La propriété de cohérence faible peut alors s'exprimer de la façon suivante :

Définition 5.40. Soit $\mathfrak{pol}$ une politique sur une topologie τ. $\mathfrak{pol}$ est dite **faiblement cohérente** si pour tout hôte de sécurité **sh** et pour toutes interfaces if, if' de $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$, $\approx_{if}^{in}$ est stable par $\overset{\mathfrak{fw}}{\longmapsto}_{if,if'}$.

On remarque que dans notre cas, seules les règles de masquerading engendrent des classes d'équivalences non triviales (c'est-à-dire comportant plus d'un élément), les classes d'équivalence se formulent donc entre paquets dont l'adresse source est une adresse issue d'une étape de masquerading et dont le port source varie dans l'intervalle des ports dynamiques. Les transformations des informations concernant les destinations de paquets n'altèrent donc pas les classes d'équivalences. Par conséquent, seules les informations relatives aux sources des packets sont d'intérêt. Considérons donc que toutes les relations définies précédemment sont restreintes aux informations relatives aux sources des paquets. Notons $\overset{\mathfrak{fw}\text{-src}}{\longmapsto}$ la relation engendrée par la succession du processus de filtrage sur les champs sources et du processus de post-routage SNAT. Pour tout couple d'interfaces if, if', $\overset{\mathfrak{fw}\text{-src}}{\longmapsto}_{if,if'}$ est une relation fortement régulière. Notons également que tout ensemble fini d'ensembles réguliers disjoints $E = \{E_1, \ldots, E_n\}$ engendre une relation d'équivalence $\approx_E$ définie par $t \approx_E t'$ ssi $t = t'$ ou s'il existe un $E_i \in E$ tel que $t, t' \in E_i$.

Proposition 5.41. Pour tout firewall $\mathfrak{fw}$ et toute interface if de $\mathfrak{fw}$, la relation d'équivalence $\approx_{if}^{in}$ est engendrée par un ensemble fini d'ensembles réguliers.

Pour le montrer, il faut et il suffit de montrer que l'identité vérifie cette propriété, que l'agrégation de relations d'équivalence (au sens du point (iv) de la définition 5.39) préserve la propriété et que la propriété est stable par composition avec une relation fortement régulière.

❝ *Démonstration.*

- La relation identité est engendrée par l'ensemble vide ;
- Soit $\approx_1, \ldots, \approx_n$ n relations engendrées par $E^1 = \{E_1^1, \ldots, E_{m_1}^1\}, \ldots, E^n = \{E_1^n, \ldots, E_{m_n}^n\}$. La plus petite relation d'équivalence $\approx$ contenant $\approx_1, \ldots, \approx_n$ est engendrée par l'ensemble fini d'ensemble réguliers constitué de l'ensemble $E^1 \cup \ldots \cup E^n$ dans lequel les ensembles non disjoints sont fusionnés (c'est-à-dire en répétant l'opération $E \cup \{E_i, E_j\} \to E \cup \{E_i \cup E_j\}$ si $E_i \cap E_j \neq \varnothing$).
- Soit $\approx$ une relation engendrée par $E = \{E_1, \ldots, E_n\}$ et R une relation binaire fortement régulière. La relation d'équivalence $\approx'$ telle que $p \approx' p'$ ssi il existe $q \approx q'$ tels que $R(q, p)$ et $R(q', p')$ est engendrée par $\{R(E_1), \ldots, R(E_n)\}$ dans lequel les ensembles non disjoints sont fusionnés.

❞

Par ailleurs, nous avons le résultat suivant :

Proposition 5.42. Soit E une relation d'équivalence sur un ensemble Ω et R une relation binaire sur $\Omega \times \Omega$. Le problème de la stabilité de E par R est décidable si E, R et Ω sont des ensembles et relations réguliers.

❝ *Démonstration.* La propriété $\forall (x,y) \in E,\ R(\{x\}) = R(\{y\})$ est équivalente à (puisque E est une relation d'équivalence) $\forall x,y,z,(E(x,y) \wedge R(x,z)) \Rightarrow R(y,z)$. Il faut et il suffit alors de vérifier que l'ensemble $\left(\sqcup_3 E \cap \sqcup_2 R \cap \sqcup_1 \overline{R}\right)$ est vide. ❞

En remarquant que toute relation d'équivalence engendrée par un ensemble fini d'ensembles réguliers est elle-même régulière et que $\overset{\mathfrak{fw}\text{-src}}{\longmapsto}_{if,if'}$ est une relation régulière, on obtient la décidabilité de la propriété de cohérence faible.

La condition suivante est une condition suffisante, simple et raisonnable de la propriété de cohérence faible : pour tout firewall $\mathfrak{fw}$ de $\mathfrak{pol}$, aucune règle de $\mathfrak{fw}$ ne contient de condition impliquant un sous-ensemble strict de l'ensemble des ports dynamiques.

Dans la suite, nous considérons une politique $\mathfrak{pol}$ faiblement cohérente. Dans ce contexte, le port utilisé par les règles de masquerading peut être considéré comme fixe. Notons p_{masq} le terme représentant le port fixe choisi parmi les ports dynamiques.

Source NAT

(S_1) $\mathbf{post}(\mathbf{sh},\mathbf{sn},x,y,z,u) \rightarrow \mathbf{forward}(\mathbf{sh},\mathbf{sn},x,y,ip',u)$
$\quad \| \quad (z,u) \in \mathcal{D}om^{\mathfrak{fw}}_{\mathrm{NAT}}(r)$
pour tout $\mathbf{sn} \in \mathsf{Net} \cup \mathsf{SH}$, $\mathbf{sh} \in \mathsf{SH}$, $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$
$r = ip \twoheadrightarrow ip' \in \mathrm{Post}_{\mathfrak{fw}}(\mu^{\mathbf{sh}}_{\mathfrak{fw}}(\mathbf{sn}))$

(S_2) $\mathbf{post}(\mathbf{sh},\mathbf{sn},x,y,z,u) \rightarrow \mathbf{forward}(\mathbf{sh},\mathbf{sn},x,y,z,port')$
$\quad \| \quad (z,u) \in \mathcal{D}om^{\mathfrak{fw}}_{\mathrm{NAT}}(r)$
pour tout $\mathbf{sn} \in \mathsf{Net} \cup \mathsf{SH}$, $\mathbf{sh} \in \mathsf{SH}$, $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$
$r = port \twoheadrightarrow port' \in \mathrm{Post}_{\mathfrak{fw}}(\mu^{\mathbf{sh}}_{\mathfrak{fw}}(\mathbf{sn}))$

(S_3) $\mathbf{post}(\mathbf{sh},\mathbf{sn},x,y,z,u) \rightarrow \mathbf{forward}(\mathbf{sh},\mathbf{sn},x,y,ip',port')$
$\quad \| \quad (z,u) \in \mathcal{D}om^{\mathfrak{fw}}_{\mathrm{NAT}}(r)$
pour tout $\mathbf{sn} \in \mathsf{Net} \cup \mathsf{SH}$, $\mathbf{sh} \in \mathsf{SH}$, $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$
$r = ip, port \twoheadrightarrow ip', port' \in \mathrm{Post}_{\mathfrak{fw}}(\mu^{\mathbf{sh}}_{\mathfrak{fw}}(\mathbf{sn}))$

(S_4) $\mathbf{post}(\mathbf{sh},\mathbf{sn},x,y,z,u) \rightarrow \mathbf{forward}(\mathbf{sh},\mathbf{sn},x,y,ip',p_{masq})$
$\quad \| \quad (z,u) \in \mathcal{D}om^{\mathfrak{fw}}_{\mathrm{NAT}}(r)$
pour tout $\mathbf{sn} \in \mathsf{Net} \cup \mathsf{SH}$, $\mathbf{sh} \in \mathsf{SH}$, $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$
$r = ip[x] \twoheadrightarrow ip' \in \mathrm{Post}_{\mathfrak{fw}}(\mu^{\mathbf{sh}}_{\mathfrak{fw}}(\mathbf{sn}))$

(S_5) $\mathbf{post}(\mathbf{sh},\mathbf{sn},x,y,z,u) \rightarrow \mathbf{forward}(\mathbf{sh},\mathbf{sn},x,y,z,u)$
$\quad \| \quad (z,u) \in \overline{\bigcup_{r \in \mathrm{Post}_{\mathfrak{fw}}} \mathcal{D}om^{\mathfrak{fw}}_{\mathrm{NAT}}(r)}$
pour tout $\mathbf{sn} \in \mathsf{Net} \cup \mathsf{SH}$, $\mathbf{sh} \in \mathsf{SH}$, $\mathfrak{fw} = \mathfrak{pol}(\mathbf{sh})$

Enfin, lorsque l'hôte de sécurité a effectué sur le packet entrant toutes les étapes qui lui incombent, le packet doit être transmis (s'il n'a pas été jeté) à l'hôte suivant (qu'il s'agisse d'un hôte de sécurité ou de l'hôte destinataire).

Forward

(FW_1) $\mathbf{forward}(\mathbf{sh}, \mathbf{net}, x, y, z, u) \rightarrow \mathbf{received}(\mathbf{net}, x, y, z, u)$
pour tout $\mathbf{net} \in \mathsf{Net}$ *et* $\mathbf{sh} = \mathcal{GW}(\mathbf{net})$

(FW_2) $\mathbf{forward}(\mathbf{sh}, \mathbf{sh}', x, y, z, u) \rightarrow \mathbf{pre}(\mathbf{sh}, \mathbf{sh}', x, y, z, u)$
pour tout $(\mathbf{sh}, \mathbf{sh}') \in \mathcal{CON}$

Exemple 5.43. Soit $\mathtt{pol}$ la politique définie dans l'exemple 5.33 ainsi que la stratégie de routage ζ définie dans l'exemple 5.35. La réduction du terme $\mathbf{sent}(\mathbf{secretariat}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(19), \mathbf{port}(515))$ par $\mathrm{Flow}_{\mathtt{pol},\zeta}$ est donnée par la figure suivante :

$$\mathbf{sent}(\mathbf{secretariat}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(19), \mathbf{port}(515))$$
$$\downarrow$$
$$\mathbf{pre}(\mathbf{secretariat}, \mathbf{sh_2}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(19), \mathbf{port}(515))$$
$$\downarrow$$
$$\mathbf{filter}(\mathbf{sh_2}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(19), \mathbf{port}(515))$$
$$\downarrow$$
$$\mathbf{route}(\mathbf{sh_2}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(19), \mathbf{port}(515))$$
$$\downarrow$$
$$\mathbf{post}(\mathbf{sh_2}, \mathbf{sh_1}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(19), \mathbf{port}(515))$$
$$\downarrow$$
$$\mathbf{forward}(\mathbf{sh_2}, \mathbf{sh_1}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(19), \mathbf{port}(515))$$
$$\downarrow$$
$$\mathbf{pre}(\mathbf{sh_2}, \mathbf{sh_1}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(19), \mathbf{port}(515))$$
$$\downarrow$$
$$\mathbf{filter}(\mathbf{sh_1}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(197), \mathbf{port}(515))$$
$$\downarrow$$
$$\mathbf{route}(\mathbf{sh_1}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(197), \mathbf{port}(515))$$
$$\downarrow$$
$$\mathbf{post}(\mathbf{sh_1}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(197), \mathbf{port}(515))$$
$$\downarrow$$
$$\mathbf{forward}(\mathbf{sh_1}, \mathbf{dpt{-}info}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(197), \mathbf{port}(515))$$
$$\downarrow$$
$$\mathbf{received}(\mathbf{dpt{-}info}, \mathbf{ip}(193), \mathbf{port}(10), \mathbf{ip}(197), \mathbf{port}(515))$$

Toutes les étapes se produisant durant l'acheminement d'un paquet depuis son émetteur jusqu'à son destinataire ont été modélisées au moyen de règles de réécriture. On peut alors définir la sémantique d'une politique de sécurité à partir de ces règles.

Définition 5.44. Étant données une politique de sécurité réseau $\mathfrak{pol}$ faiblement cohérente sur une topologie τ et une stratégie de routage correcte pour τ, on appelle **sémantique de $\mathfrak{pol}$ relativement à** ζ et l'on note $[\![\mathfrak{pol}]\!]_\zeta$ la relation (partielle ou totale) associant à tout $r \in \mathcal{R}equest$ la ou les décisions $d \in \mathcal{D}ecision$, lorsqu'elles existent, telles que $r \xrightarrow{*}_{\mathrm{Flow}_{\mathfrak{pol},\zeta}} d$ où $\mathrm{Flow}_{\mathfrak{pol},\zeta}$ est le système de réécriture défini par les règles (B), (G), (D_1), (D_2), (D_3), (D_4), (F_1), (F_2), (R_1), (R_2), (S_1), (S_2), (S_3), (S_4), (S_5), (FW_1), (FW_2).

Notons que par construction (ce qui est conforme à la réalité) et eu égard les conditions imposées sur ζ, le système de réécriture $\mathrm{Flow}_{\mathfrak{pol},\zeta}$ est déterministe en ce sens où pour tout terme clos t, il existe au plus un terme t' tel que $t \to_{\mathrm{Flow}_{\mathfrak{pol},\zeta}} t'$.

5.4.5 Propriétés et analyse

Dans cette section, nous nous intéressons à quelques propriétés essentielles et proposons de montrer comment la spécification des politiques au moyen d'un système de réécriture nous permet de raisonner sur ces propriétés.

Comme nous l'avons déjà mentionné, la complétude est l'aptitude à prendre une décision quelle que soit la requête d'accès formulée. Dans le cas d'un firewall isolé, la vérification de cette propriété s'effectue simplement en vérifiant que la première composante de la sémantique de ce dernier (en tant que relation régulière) couvre tous les paquets possibles. Pour autant, le cas des politiques de sécurité réseau est plus complexe. Ceci est dû au fait que l'incomplétude peut être engendrée par diverses raisons. Un firewall incomplet peut en être la cause mais le fait qu'un paquet ne soit jamais jeté ni délivré peut également se produire (auquel cas on dit que le paquet est perdu).

Définition 5.45. Une politique de sécurité $\mathfrak{pol}$ sur une topologie τ est dite **complète** relativement à la stratégie de routage ζ, ou encore ζ-*complète*, si pour toute requête $r \in \mathcal{R}equest$, il existe une décision $d \in \mathcal{D}ecision$ telle que $[\![\mathfrak{pol}]\!]_\zeta(r) = d$.

La forme particulière du système de réécriture nous permet d'énoncer la propriété suivante :

Proposition 5.46. Pour tout politique de sécurité $\mathfrak{pol}$ et toute stratégie de routage ζ, la propriété de ζ-complétude de $\mathfrak{pol}$ est décidable.

Plus précisément, nous avons la propriété suivante :

Proposition 5.47. Étant donnée une politique de sécurité réseau $\mathfrak{pol}$ sur une topologie τ ainsi qu'une stratégie de routage ζ, les ensembles :

i) $(\to^*_{\mathrm{Flow}_{\mathfrak{pol},\zeta}})^{-1}(\{\mathbf{dropped}\})$ et

ii) $(\to^*_{\mathrm{Flow}_{\mathfrak{pol},\zeta}})^{-1}(\mathcal{D}ecision \setminus \{\mathbf{dropped}\})$

sont effectivement réguliers.

❝ *Démonstration.* Les preuves des propositions 5.46 et 5.47 sont basées sur le fait que $\text{Flow}_{\mathfrak{pol},\zeta}$ est un système de réécriture contraints croissant à contraintes régulières. Il est proposé dans [Jacquemard, 1996b] une méthode pour calculer l'automate d'arbre reconnaissant l'ensemble $(\to_{\mathrm{R}}^*)^{-1}(L)$ pour tout ensemble régulier L et tout système de réécriture croissant R. Montrons que l'ajout de contraintes régulières n'altère pas ce résultat. Sans perte de généralité, nous pouvons considérer que tout SRC-GRS est un ensemble de règles de l'une des formes suivantes :

$$x \to r[x] \quad \| \quad x \in A \tag{5.1}$$

$$f(x_1, \ldots, x_n) \to r[x_1, \ldots, x_n] \quad \| \quad x_1 \in A_1, \ldots, x_n \in A_n \tag{5.2}$$

sachant que toute variable non contrainte x peut être vue comme une variable contrainte par $x \in A$ où A est l'automate reconnaissant tous les termes clos de la sorte de x. En conséquence, nous pouvons considérer que toutes les variables sont contraintes. Soit L un langage régulier reconnu par A_L et R un SRC-GRS. L'automate reconnaissant $(\to_{\mathrm{R}}^*)^{-1}(L)$ peut être construit de la façon suivante :

$$\mathcal{R}each_0 = (Q, Q_0, \Delta_0) := \biguplus_{(l \to r \| C) \in \mathrm{R}} \left(\biguplus_{(x \in A) \in C} A \right) \uplus A_L$$

où la somme disjointe ($\uplus$) de deux automates sur une même signature est l'automate dont l'ensemble des états, des états finaux et des règles sont respectivement l'union des ensembles correspondants des deux automates, étant entendu que ces derniers sont tous disjoints. Ainsi, l'ensemble de règle Δ_k ($k \geq 0$) est transformé en l'ensemble Δ_{k+1} en appliquant les règles suivantes :

$$(i) \ \frac{\begin{cases} (f(x_1, \ldots, x_n) \to g(r_1, \ldots, r_m) \| \bigwedge_i x_i \in A_i) \in \mathrm{R} \\ g(q_1, \ldots, q_m) \to q \in \Delta_k \end{cases}}{f(q'_1, \ldots, q'_n) \to q \in \Delta_{k+1}}$$

sous les conditions :

1. pour tout $1 \leq i \leq n$, q'_i est un état final de A_i,
2. pour tout $1 \leq j \leq m$, il existe une substitution $\theta : \mathcal{X} \to Q$ telle que $\theta(r_j) \xrightarrow{*}_{\Delta_k} q_j$ et pour chaque x_i apparaissant dans $g(r_1, \ldots, r_m)$, on ait $\theta(x_i) = q'_i$.

$$(ii) \ \frac{(f(x_1, \ldots, x_n) \to x \| \bigwedge_i x_i \in A_i) \in \mathrm{R} \ \ ; \ \ q \in Q}{f(q'_1, \ldots, q'_n) \to q \in \Delta_{k+1}}$$

sous les conditions :

1. x est une variable,
2. pour tout $1 \leq i \leq n$, si $x_i = x$ alors $q'_i = q$, sinon q'_i est un état final de A_i.

L'automate désiré est (Q, Q_L, Δ) où Q_L est l'ensemble des états finaux de A_L.

Ainsi, et puisque $\{\mathbf{dropped}\}$ et $\mathcal{D}ecision \setminus \{\mathbf{dropped}\}$ sont des ensembles réguliers, il est possible de construire l'automate reconnaissant les ensembles $(\to_{\text{Flow}_{\mathfrak{pol},\zeta}}^*)^{-1}(\{\mathbf{dropped}\})$ et $(\to_{\text{Flow}_{\mathfrak{pol},\zeta}}^*)^{-1}(\mathcal{D}ecision \setminus \{\mathbf{dropped}\})$. La complétude d'une politique de sécurité peut ainsi être vérifiée en testant que l'union de ces deux automates est égal à l'ensemble régulier $\mathcal{R}equest$ de toutes les requêtes. ❞

Même si une stratégie de routage est correcte, la présence de règles de traduction d'adresses peut engendrer des boucles dans l'acheminement d'un paquet. En effet, certains paquets peuvent être « piégés » dans une boucle impliquant plusieurs étapes NAT « contradictoires ». Par exemple, un hôte **sh** peut transformer un paquet p en p' et le transmettre à $\mathbf{sh}'$ qui peut éventuellement traduire p' en p avant de le re-transmettre à **sh**. De telles boucles peuvent impacter significativement la qualité du trafic voire engendrer un déni de service. Nous dirons qu'une stratégie de routage est *sûre* sous une politique de sécurité donnée si la sémantique de la politique relativement à cette stratégie n'engendre aucune boucle. On s'aperçoit par ailleurs que la taille d'un terme donné est bornée au cours du processus de réécriture par $\mathrm{Flow}_{\mathfrak{pol},\zeta}$. En conséquence, seules les boucles peuvent être à l'origine de la non-terminaison du système de réécriture. Ainsi, on obtient la proposition suivante :

Proposition 5.48 (Routage sûr). Étant donnée une politique de sécurité $\mathfrak{pol}$ sur une topologie τ, une stratégie de routage ζ est *sûre* sous la politique $\mathfrak{pol}$ ssi $\mathrm{Flow}_{\mathfrak{pol},\zeta}$ termine.

Nous considérions jusqu'ici des politiques de sécurité relativement à des stratégies de routage statiques. Cependant, en pratique, seuls les petits réseaux configurent manuellement les tables de routage. La plupart du temps la topologie des réseaux est trop complexe ou trop muable, rendant la définition d'une stratégie de routage statique impossible. De surcroît, avec la complexité grandissante des réseaux, les notions de performance et d'efficacité du processus de routage émergent et dans ce cadre, il est préférable de calculer dynamiquement le chemin d'un paquet en fonction de la situation du trafic et de déterminer le meilleur chemin, à savoir le plus court, le plus sûr ou encore celui qui équilibre le mieux la charge du réseau. C'est pourquoi le routage adaptatif, ou routage dynamique, est largement utilisé.

Dans ce contexte, il est crucial que le sort réservé à chaque paquet ne dépende pas de la route qu'il emprunte. Autrement dit, un paquet doit soit être jeté, soit être délivré au même destinataire dans tous les cas de figure. En d'autres termes, le routage doit être une stratégie pour atteindre un certain but mais ne doit pas altérer ce but. On appelle, comme précédemment, « cohérence » la propriété assurant que la sémantique d'une politique de sécurité est la même quelle que soit la stratégie de routage (correcte) considérée.

Définition 5.49. Une politique de sécurité $\mathfrak{pol}$ sur une topologie τ est dite **cohérente** ssi pour toutes stratégies de routage ζ et ζ' correcte relativement à τ, $[\![\mathfrak{pol}]\!]_\zeta = [\![\mathfrak{pol}]\!]_{\zeta'}$.

Nous exprimons la cohérence d'une politique comme une propriété sur un nouveau système de réécriture. Étant donnée une politique $\mathfrak{pol}$, nous définissons $\mathrm{Flow}_{\mathfrak{pol}}$ comme étant le système de réécriture linéaire à gauche décrivant tous les scénarios d'acheminement possibles relativement aux stratégies de routage correctes, ou, autrement dit, comme le système transformant toute requête r en une décision $d \in \mathcal{D}ecision$ ssi il existe une stratégie de routage correcte ζ telle que $[\![\mathfrak{pol}]\!]_\zeta(r) = d$.

Pour construire le système de réécriture $\text{Flow}_{\mathfrak{pol}}$, nous ajoutons à la signature Σ_τ les symboles suivants :

$$\begin{array}{rcl} \bot, \top & : & \to \mathsf{Bool} \\ \langle _, \ldots, _ \rangle_ & : \overbrace{\mathsf{Bool} \times \ldots \times \mathsf{Bool}}^{m \text{ occurences}} \times \mathsf{SH} & \to \mathsf{Context} \\ :: & : \mathsf{Context} \times \mathsf{Trace} & \to \mathsf{Trace} \\ \# & : & \to \mathsf{Trace} \end{array}$$

où m est le nombre d'hôtes de sécurité et modifions les autres symboles en remplaçant dans chaque profil la sorte SH par la sorte Trace. Un terme de la forme $\langle\, t_1, \ldots, t_n \,\rangle_{\mathbf{sh}}$ (un contexte) signifie que dans l'état courant, l'hôte de sécurité $\mathbf{sh}_i$ a été transersé par le paquet (dans sa forme courante) ssi $t_i = \top$. Un terme de sorte Trace représente la pile de parcours du paquet. Étant donnée une politique de sécurité $\mathfrak{pol}$, nous définissons $\text{Flow}_{\mathfrak{pol}}$ en modifiant le système $\text{Flow}_{\mathfrak{pol},\zeta}$ de la façon suivante : pour toutes les règles suivantes, les contraintes sont inchangées et nous n'illustrons que le sous-terme correspondant à la sorte Trace des membres des règles :

- (G) : $\mathbf{sh}$ est remplacé dans le membre droit par $\langle\, \bot, \ldots, \underbrace{\top}_{\text{indice de } \mathbf{sh}}, \ldots, \bot \,\rangle_{\mathbf{sh}} :: \#$ indiquant qu'à l'initialisation, seule la passerelle a inspecté le paquet ;
- $(D_4), (S_5), (F_j), (R_i), (FW_1)$: $\mathbf{sh}$ est remplacé par $\langle\, x_1, \ldots, x_n \,\rangle_{\mathbf{sh}}$;
- $(D_1), (D_2), (D_3), (S_1), (S_2), (S_3), (S_4)$: $\mathbf{sh}$ est remplacé dans le membre gauche par $\langle\, x_1, \ldots, x_m \,\rangle_{\mathbf{sh}} :: q$ et dans le membre droit par $\langle\, \bot, \ldots, \underbrace{\top}_{\text{indice de } \mathbf{sh}}, \ldots, \bot \,\rangle_{\mathbf{sh}} :: \#$, indiquant que la trace du parcours est réinitialisée lorsqu'un paquet est modifié (puisque du point de vue du réseau, il ne s'agit plus du même paquet) ;
- (FW_2) : $\mathbf{sh}$ est remplacé dans le membre gauche par

$$\langle\, x_1, \ldots, \underbrace{\bot}_{\text{indice de } \mathbf{sh}'}, \ldots, x_m \,\rangle_{\mathbf{sh}} :: q$$

et dans le membre droit par

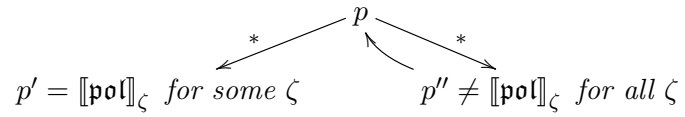
$$\langle\, x_1, \ldots, \underbrace{\top}_{\text{indice de } \mathbf{sh}'}, \ldots, \underbrace{\top}_{\text{indice de } \mathbf{sh}}, \ldots, x_m \,\rangle_{\mathbf{sh}'} :: \langle\, x_1, \ldots, \underbrace{\top}_{\text{indice de } \mathbf{sh}'}, \ldots, x_m \,\rangle_{\mathbf{sh}} :: q$$

indiquant que si un retour-arrière est effectué sur $\mathbf{sh}$, alors $\mathbf{sh}'$ ne devra plus faire parti des choix de routage ;

et la règle suivante est ajoutée :

$$(R_{back}) \quad \mathbf{route}(\langle\, \underbrace{x_1, \ldots, x_m}_{x_i = \top \text{ pour tout } (\mathbf{sh}, \mathbf{sh_i}) \in \mathcal{CON}} \,\rangle_{\mathbf{sh}} :: q, x, y, z, u) \to \mathbf{route}(q, x, y, z, u)$$

Grossièrement, la pile sert à simuler le choix non déterministe de toutes les routes possibles pour essayer d'atteindre une décision. Si un chemin en cours d'exploration ne conduit pas à une décision, alors le terme effectue un retour-arrière :



Pour ne pas engendrer de cycle dans le processus d'exploration, les paquets sont étiquetés par l'ensemble des branches déjà explorées. De cette façon, l'on obtient un système de réécriture transformant toute requête en toutes les décisions possibles correspondant à une stratégie correcte de routage. Plus précisément, on obtient un système vérifiant la proposition suivante :

Proposition 5.50. Soit $\mathfrak{pol}$ une politique de sécurité réseau sur une topologie τ. $\text{Flow}_{\mathfrak{pol}}$ termine ssi toute stratégie de routage correcte est sûre pour la politique $\mathfrak{pol}$. Dans ce cas, $\mathfrak{pol}$ est cohérent ssi $\text{Flow}_{\mathfrak{pol}}$ est confluent sur les termes clos.

❝ *Démonstration.* On peut facilement montrer que $r \xrightarrow{*}_{\text{Flow}_{\mathfrak{pol}}} d \in \mathcal{D}ecision$ ssi il existe une stratégie ζ telle que $[\![\mathfrak{pol}]\!]_\zeta(r) = d$. Partant de cela, l'équivalence entre la cohérence de $\mathfrak{pol}$ et la confluence sur les termes clos de $\text{Flow}_{\mathfrak{pol}}$ est établie ssi pour toutes stratégies de routage ζ et ζ', $\mathcal{D}om([\![\mathfrak{pol}]\!]_\zeta) = \mathcal{D}om([\![\mathfrak{pol}]\!]_{\zeta'})$. Puisque l'on considère uniquement des stratégies correctes, lorsqu'une requête ne peut pas être évaluée, deux cas seulement sont possibles : soit la requête implique une adresse non-routable, soit la stratégie n'est pas sûre. Dans le premier cas, la requête ne peut pas être résolue quelle que soit la stratégie. Ainsi, si toutes les stratégies sont sûres, elles ont nécessairement le même domaine. ❞

Exemple 5.51. Considérons la politique décrite dans l'exemple 5.33. Nous avons montré de façon automatique (en utilisant CiME [Contejean et al., 2010] et AProVE [Giesl et al., 2006]) que $\text{Flow}_{\mathfrak{pol}}$ est un système terminant et en conséquence que toute stratégie correcte est sûre pour $\mathfrak{pol}$. Nous avons également tenté de montrer la cohérence de $\mathfrak{pol}$ mais la preuve a échoué : la dérivation suivante prouvant que $\mathfrak{pol}$ n'est pas cohérent :

$$\begin{array}{c}
\mathbf{sent}(\mathbf{secretariat}, \mathbf{ip}(195), \mathbf{port}(32), \mathbf{ip}(49), \mathbf{port}(25)) \\
\downarrow^* \\
\mathbf{fwd}(\langle \bot, \top, \bot \rangle_{\mathbf{sh_2}} :: \#, \mathbf{sh_3}, \mathbf{ip}(195), \mathbf{port}(32), \mathbf{ip}(49), \mathbf{port}(25)) \\
\downarrow^* \\
\mathbf{filter}(\langle \bot, \top, \top \rangle_{\mathbf{sh_3}} :: \ldots, \mathbf{ip}(195), \mathbf{port}(32), \mathbf{ip}(49), \mathbf{port}(25)) \\
\downarrow^* \\
\mathbf{dropped} \\
\downarrow \\
\mathbf{post}(\langle \top, \bot, \bot \rangle_{\mathbf{sh_1}} :: \ldots, \mathbf{sh_2}, \mathbf{ip}(195), \mathbf{port}(32), \mathbf{ip}(49), \mathbf{port}(25)) \\
\downarrow^* \\
\mathbf{fwd}(\langle \top, \bot, \bot \rangle_{\mathbf{sh_1}} :: \ldots, \mathbf{sh_2}, \mathbf{ip}(17), \mathbf{port}(8080), \mathbf{ip}(49), \mathbf{port}(25)) \\
\downarrow^* \\
\mathbf{fwd}(\langle \top, \top, \bot \rangle_{\mathbf{sh_2}} :: \ldots, \mathbf{sh_3}, \mathbf{ip}(17), \mathbf{port}(8080), \mathbf{ip}(49), \mathbf{port}(25)) \\
\downarrow^* \\
\mathbf{filter}(\langle \top, \top, \top \rangle_{\mathbf{sh_3}} :: \ldots, \mathbf{ip}(17), \mathbf{port}(8080), \mathbf{ip}(49), \mathbf{port}(25)) \\
\downarrow^* \\
\mathbf{received}(\mathbf{servers}, \mathbf{ip}(17), \mathbf{port}(8080), \mathbf{ip}(49), \mathbf{port}(25))
\end{array}$$

Cette dérivation montre que, sous une stratégie de routage spécifique, les hôtes

> du réseau **secretariat** peuvent exploiter une faille de la politique de l'hôte de sécurité $\mathbf{sh_1}$ pour attaquer le serveur **servers**.

5.5 Travaux reliés

La littérature concernant le développement de méthodes et outils pour analyser et tester les politiques de sécurité dans les réseaux est relativement abondante. Les méthodes existantes sont partitionnées en deux catégories : les méthodes dites actives et celles dites passives. Les premières consistent à envoyer des paquets dans le réseau pour effectuer un diagnostic à la lumière des paquets reçus. L'avantage incontestable de cette classe de méthode est qu'elle ne nécessite aucune représentation abstraite des firewalls, en conséquence de quoi aucune erreur de modélisation ni simplification ne peut être introduite, validant de façon certaine le résultat de l'analyse effectuée. Cependant, il est évident que ces méthodes possèdent l'inconvénient majeur de consommer de la bande passante, d'interférer avec le traffic et de ne permettre qu'une analyse très partielle de la situation. C'est la raison pour laquelle ont émergées depuis quelques années les méthodes dites passives. Ces méthodes consistent à effectuer une analyse « hors ligne » de la politique de sécurité. C'est dans cette démarche que s'inscrivent les travaux relatés dans ce chapitre.

Parmi les méthodes passives, deux principales approches ont été investiguées dans la littérature : les analyses structurelles et les analyses par requêtage. Un certain nombre de travaux (notamment [Cuppens et al., 2006, Abbes et al., 2008, Benelbahri and Bouhoula, 2007, Gouda and Liu, 2004, Cuppens et al., 2005] [Liu, 2008, Al-Shaer et al., 2005]) se sont attachés à développer les méthodes d'analyse structurelles en se focalisant notamment sur la définition, la détection et la discussion d'anomalies tandis que d'autres (particulièrement [Hazelhurst, 2000] [Eronen and Zitting, 2001, Liu and Gouda, 2009]) se sont concentrés sur les méthodes permettant d'effectuer des analyses par requêtage. La plupart de ces travaux abstraient les règles de filtrage par des règles comportant une ou deux dimensions d'intervalles d'entiers, ce qui limite l'intérêt des résultats obtenus. D'autres émettent l'hypothèse que les paquets ne sont pas modifés durant leur traversée du réseau et, en conséquence, ne prennent pas en compte les règles de traduction d'adresse pourtant omniprésentes dans les réseaux contemporains. De plus, les approches proposées se focalisent principalement sur l'étude des firewalls isolés. Les approches proposées traitent divers aspects (une comparaison détaillée entre les différentes techniques a été établie dans [Nelson et al., 2010]) mais les aspects relatifs à la topologie et au routage ne sont pas étudiés en profondeur. En somme, les analyses sont restreintes aux politiques locales et leur combinaison est peu ou pas étudiée. Parmi les méthodes basées sur une spécification algébrique des firewalls, on compte principalement les travaux initiés dans [Kirchner et al., 2009]. Dans cet article, les firewalls sont spécifiés par des systèmes de réécriture et l'analyse par requêtage des firewalls isolés est effectuée via les techniques de narrowing. Cette approche constitue une première étape vers l'établissement d'une méthode plus complète et plus dédiée pour l'analyse des politiques de sécurité dans les réseaux via les systèmes de réécriture. En ce sens, les travaux présentés dans ce chapitre s'inscrivent dans une démarche similaire.

5.6 Synthèse

Nous avons proposé dans ce chapitre une approche pour décrire les politiques de sécurité réseaux sous la forme de systèmes de réécriture et d'automates d'arbres. Nous avons montré que cette approche peut être utilisée pour effectuer plusieurs types d'analyse sur les firewalls isolés ainsi que sur les combinaisons complexes de firewalls. Dans ces travaux, nous nous sommes attaché à émettre des hypothèses de modélisation conformes à la réalité (sur les processus de filtrage et de traduction d'adresses). Nous avons montré que les spécifications des politiques de sécurité réseau proposées sont adaptées pour vérifier les propriétés usuelles liées à la sémantique des firewalls et pour les comparer entre eux. Elles permettent également de spécifier et de détecter automatiquement les anomalies de configuration et d'effectuer des analyses par requêtage. De plus, toutes ces analyses sont effectuées dans un formalisme commun. La même approche a été utilisée pour spécifier les politiques de sécurité des réseaux, tenant compte de leurs topologies et des stratégies de routage utilisées. Le système de réécriture obtenu fournit une spécification exécutable du trafic réseau soumis à la politique de sécurité considérée et, grâce à la forme particulière de ses règles, peut être utilisée pour effectuer des vérifications de complétude, d'accessibilité, de terminaison et de cohérence. Certaines de ces propriétés, comme la cohérence, sont considérées relativement à une stratégie de routage donnée ou pour toutes les stratégies possibles.

Les principaux résultats présentés dans ce chapitre ont fait l'objet de deux publications : [Bourdier, 2011b, Bourdier and Cirstea, 2011].

6 Synthèse

L'analyse formelle des politiques de sécurité et des mécanismes les mettant en oeuvre permet d'accroître la qualité et la sûreté de la sécurité des systèmes d'information et des réseaux. Les travaux de thèse décrits dans ce manuscrit s'inscrivent dans une démarche générale d'établissement d'un cadre permettant la définition de politiques de sécurité et la vérification de leurs propriétés.

6.1 Contributions

Un cadre formel pour la spécification, l'analyse et la transformation de politiques de sécurité statiques

Dans une première partie, nous avons proposé un cadre pour la spécification de politiques de sécurité basé sur une approche modulaire dans laquelle une politique est vue comme la composition d'un modèle de sécurité et d'une configuration. Nous avons investigué les possibilités offertes par de telles spécifications lorsque les modèles sont exprimés au moyen de contraintes du premier ordre et les configurations au moyen de programmes logiques $\mathcal{H}_1$. Nous avons démontré l'adéquation d'un tel cadre pour analyser les propriétés des politiques ainsi définies. En particulier, nous avons proposé un algorithme permettant de transformer une politique exprimée dans un modèle donné vers une autre politique équivalente (au sens où elle engendre les mêmes autorisations) exprimée dans un autre modèle. Une précédente version de ces travaux a été présentée dans [Bourdier, 2011a].

Un cadre formel pour la spécification et l'analyse de politiques de sécurité dynamiques

Dans un second temps, nous nous sommes proposé de tenir compte des aspects dynamiques de la configuration d'une politique vue comme un état du système sur lequel la politique est mise en oeuvre et où chaque action est associée à une procédure de modification des états. Nous avons proposé un langage simple pour spécifier séparément les systèmes et les politiques de sécurité. La mise en place d'un langage « dédié » a permis d'imposer des restrictions syntaxiques complexes sur la logique sous-jacente (les systèmes de réécriture) de façon transparente pour l'utilisateur. En ce sens, notre approche se distingue fortement de la plupart des approches formelles proposées pour la spécification des politiques dans la littérature, nécessitant la maîtrise du formalisme utilisé par l'administrateur de sécurité. Nous avons ensuite donné une sémantique des spécifications exprimées dans ce cadre sous la forme de systèmes de réécriture. Nous nous sommes ensuite attaché à montrer que les systèmes de réécriture obtenus permettent l'étude de propriétés de sécurité. En somme, nous avons proposé une démarche reposant sur des langages différents combinés puis traduits dans un formalisme de raisonnement commun. Une partie de ces travaux a été présentée dans [Bourdier et al., 2010b, Bourdier et al., 2010c, Bourdier et al., 2011].

Formalisation et analyse des politiques de sécurité dans les réseaux informatiques

Dans une troisième partie, nous nous sommes focalisé sur les mécanismes permettant la mise en oeuvre de politiques de sécurité dans les réseaux. Dans ce cadre, nous avons proposé une spécification des firewalls et de leurs compositions basée sur les automates d'arbres et les systèmes de réécriture puis avons montré en quoi ces spécifications nous permettent d'analyser de façon automatique les politiques de sécurité sous-jacentes. Nous avons notamment étendu les travaux existants en proposant un cadre unifié pour effectuer des analyses usuelles traitées dans différents formalismes et en intégrant les notions, souvent négligées, de routage et de topologie. Les principaux résultats présentés dans ce chapitre ont fait l'objet de deux publications : [Bourdier, 2011b, Bourdier and Cirstea, 2011].

Autres contributions

Une partie du travail mené durant cette thèse n'a pas été présentée dans ce manuscrit car issue d'investigations non abouties au regard des problématiques de sécurité. Ce travail est relatif à l'étude des stratégies de réécriture. L'étude menée dans le chapitre 4 a présenté les politiques de sécurité comme un moyen de contrôler l'évolution d'un système de transitions. Cette vision des politiques est similaire à une interprétation possible de la notion de stratégie de réécriture. Nous avons donc investigué la formalisation d'un cadre dans lequel les systèmes sont décrits au moyen de règles de réécriture et les politiques au moyen de stratégies. La formalisation usuelle des stratégies permettait assez mal l'analyse des problèmes liés à leur utilisation dans ce cadre. Nous avons donc proposé une reformulation de la définition des stratégies [Bourdier et al., 2010a], d'un point de vue abstrait comme concret (constructif) te-

nant compte de la diversité des types de stratégies révélés par l'analyse des politiques de sécurité ; les stratégies habituellement étudiées étant principalement structurelles, il a fallu permettre la formalisation de stratégies tenant compte d'autres variables environnementales, engendrant un changement radical dans l'approche de ces dernières. À cet effet, un langage permettant la construction de stratégies a été proposé et un prototype de compilateur réalisé, permettant une première analyse des stratégies nouvellement définies.

6.2 Perspectives

Décidabilité de la transformation des politiques statiques

La prochaine étape faisant suite aux travaux décrits dans le chapitre 3 consiste à déterminer la classe de modèles pour lesquelles l'existence d'une politique équivalente est décidable. En effet, dans l'état actuel de nos travaux, le processus de transformation que nous avons établi peut retourner, pour une entrée donnée, soit une configuration répondant aux exigences formulées, soit l'information indiquant qu'aucune configuration répondant aux contraintes n'existe, soit ne pas terminer. Il est donc pertinent d'étudier les conditions sous lesquelles le processus de transformation termine toujours.

Politique abstraite

Il arrive souvent que les besoins de sécurité s'expriment en fonction d'éléments qui ne sont pas directement présents dans le système que l'on étudie. C'est notamment le cas lorsque l'on établit une politique sur les flots d'information alors que seule la notion d'accès est présente dans le système. Cette différence de vocabulaire entre le domaine de discours de la politique de sécurité et celui du système entraine la nécessité de mettre en oeuvre une passerelle entre ces deux univers. Dans [Bourdier et al., 2011], nous avons entrevu la notion de transformation d'environnements permettant de spécifier les besoins de sécurité dans un langage différent de celui du système. Si le cadre proposé permet d'exprimer tous les ingrédients nécessaire au raisonnement, il n'en permet pas l'automatisation. Il s'agirait alors d'établir un langage permettant l'expression de l'interprétation des éléments de la politique en fonction des données du système et de permettre, soit de dériver automatiquement une politique de sécurité s'exprimant sur le vocabulaire du système, soit d'en vérifier la conformité.

Composition de politiques

La plupart des techniques de vérification sont connues pour leur difficile passage à l'échelle dès lors que la taille des systèmes étudiés devient importante ; savoir composer et décomposer preuves et systèmes semble donc nécessaire pour appliquer avec succès les méthodes de vérification formelles. Ce besoin de modularité est d'autant plus essentiel dans un contexte où les systèmes sont de plus en plus hétérogènes ou distribués. Il s'agit donc, selon nous, d'un axe de recherche déterminant pour la communauté des méthodes formelles. Ce besoin de modularité dans la spécification et

la vérification est l'une des conclusions de ces travaux de thèse. Il s'agit par ailleurs d'un état de fait bien connu de la communauté de « réécriture » : de nombreux travaux se sont attachés à déterminer des approches modulaires dans les preuves de terminaison et de confluence des systèmes de réécriture de termes. Dans ce contexte, il semble pertinent de poursuivre les efforts de modularité effectués dans nos travaux pour proposer, à terme, un cadre permettant la composition de politiques. La notion de composition peut, par ailleurs, être envisagée de multiples façons. Une composition, que l'on pourrait qualifier d'horizontale, s'envisage comme la combinaison de politiques agissant sur le même système au moyen d'opérateurs. Une seconde composition, que nous qualifierions de verticale, s'envisage comme l'application d'une politique dite administrative sur une autre politique. Ce type de composition s'apparenterait à une notion de politique « d'ordre supérieur ».

Extension des possibilités d'analyses des politiques de sécurité réseaux

Plusieurs directions de prolongement des travaux présentés dans le chapitre 5 sont envisageables. Il pourrait, par exemple, être intéressant d'utiliser les techniques de (dés)unification [Jouannaud and Kirchner, 1991] pour effectuer sur une politique de sécurité réseau des analyses similaires à celles effectuées pour les firewalls isolés. Ceci permettrait, par exemple, de détecter les canaux cachés en résolvant, modulo le système de réécriture $\mathrm{Flow}_{\mathtt{pol}}$, des équations de la forme :

$$\begin{aligned} & \mathbf{sent}(n, \mathbf{from}(x, x'), \mathbf{dest}(y, y')) = \mathbf{dropped} \\ \wedge\ & \mathbf{sent}(n, \mathbf{from}(x_1, x_2), \mathbf{dest}(y_1, y_2)) \\ & \qquad = \mathbf{received}(n', \mathbf{from}(x, x'), \mathbf{dest}(y, y')) \end{aligned}$$

Bien évidement, pour utiliser les techniques et outils spécifiques aux systèmes de réécriture, il est nécessaire de pouvoir représenter le système $\mathrm{Flow}_{\mathtt{pol}}$ sous la forme d'un système non contraint. La forme particulière des contraintes nous permet de le faire sans augmenter la signature ni même altérer les propriétés de linéarité du système contraint. Ceci est dû au fait que les ensembles réguliers des contraintes des règles de réécriture sont basés sur des langages préfixes. Chacun des automates des contraintes considérés reconnaît l'ensemble des instances closes d'un ensemble fini de termes linéaires. On peut donc aisément transformer chaque règle contrainte en règle non contrainte. On obtient alors un système de réécriture linéaire contenant seulement des superpositions triviales (en revanche, on perd le caractère croissant du système).

Modélisation d'autres mécanismes de sécurité pour la mise en oeuvre des politiques dans les réseaux

Une autre perspective envisageable est d'étendre notre approche pour prendre en compte les firewalls à états et de modéliser le principe de proxy. Plus précisément, pour modéliser complètement le principe des connections TCP, nous devrons considérer les requêtes d'accès non plus comme des paquets mais comme des séquences de paquets. Ceci nous permettra de capturer de nouvelles vulnérabilités comme par exemple les dénis de services, provoquées par une non-synchronisation de firewalls.

Proxy

Un proxy, aussi appelé « serveur mandataire », est un hôte destiné à servir d'intermédiaire entre deux hôtes (un client et un serveur) en relayant les connections. Lorsqu'un proxy reçoit un paquet, il émet un nouveau paquet au regard des informations contenues dans le paquet entrant.

À court terme, nous envisageons d'intégrer la modélisation des proxies. Dans la mesure où le principe de leur fonctionnement réside essentiellement en la réécriture de séquences de paquets, notre approche semble parfaitement adaptée à la modélisation de ces derniers.

Enfin, il semble pertinent d'offir une vision plus « haut niveau » des politiques de sécurité réseaux en intégrant la notion de services. Les politiques pourraient alors être formulées en terme d'accès des utilisateurs à des services. L'étude de la conformité d'une politique de sécurité ainsi définie avec sa mise en oeuvre sous la forme d'une combinaison de firewalls constituerait une suite logique de ces travaux.

Bibliographie

[Abadi and Fournet, 2003] Abadi, M. and Fournet, C. (2003). Access control based on execution history. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium*, pages 107–121. Internet Society.

[Abbes et al., 2008] Abbes, T., Bouhoula, A., and Rusinowitch, M. (2008). An inference system for detecting firewall filtering rules anomalies. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 2122–2128. ACM.

[Abdulla et al., 2002] Abdulla, P., Jonsson, B., Mahata, P., and d'Orso, J. (2002). Regular tree model checking. In *Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 452–466. Springer-Verlag.

[Abiteboul et al., 1995] Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of databases*. Addison-Wesley.

[Abrams et al., 1995] Abrams, M., Jajodia, S., and Podell, H., editors (1995). *Sécurité des systèmes d'information*. IEEE Computer Society.

[Adiseshu et al., 2000] Adiseshu, H., Suri, S., and Parulkar, G. (2000). Detecting and resolving packet filter conflicts. In *INFOCOM 2000 - 9th annual joint Conference of the IEEE Computer and Communication Societies*, volume 3, pages 1203–1212. IEEE Computer Society.

[Al-Shaer and Hamed, 2003] Al-Shaer, E. and Hamed, H. (2003). Firewall policy advisor for anomaly detection and rule editing. In *IFIP/IEEE 8th International Symposium on Integrated Network Management*, volume 246 of *IFIP Conference Proceedings*, pages 17–30. Kluwer Academic Publishers.

[Al-shaer and Hamed, 2004] Al-shaer, E. and Hamed, H. (2004). Discovery of policy anomalies in distributed firewalls. In *INFOCOM 2004 - 23rd annual joint Conference of the IEEE Computer and Communication Societies*, volume 4, pages 2605–2616. IEEE Computer Society.

[Al-Shaer et al., 2005] Al-Shaer, E., Hamed, H., Boutaba, R., and Hasan, M. (2005). Conflict classification and analysis of distributed firewall policies. *IEEE Journal on Selected Areas in Communications*, 23(10) :2069 – 2084.

[Alferes et al., 2002] Alferes, J., Pereira, L., Przymusinska, H., and Przymusinski, T. (2002). LUPS – A language for updating logic programs. *Artificial Intelligence*, 138(1-2) :87–116. Elsevier Science Publishers B. V.

[ANSSI, 2011] ANSSI (2011). Portail de la sécurité informatique – agence nationale de la sécurité des systèmes d'information. http://www.ssi.gouv.fr/.

[Armando et al., 2005] Armando, A. et al. (2005). The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In Etessami, K. and Rajamani, S., editors, *Computer Aided Verification*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer-Verlag.

[Avolio, 1999] Avolio, F. (1999). Firewalls and Internet security, the second hundred (Internet) years. *The Internet Protocol Journal*, 2(2) :24–32. Ole Jacobsen publisher, CISCO.

[Baader and Nipkow, 1998] Baader, F. and Nipkow, T. (1998). *Term rewriting and all that*. Cambridge University Press. Cambridge University Press.

[Bachmair and Dershowitz, 1989] Bachmair, L. and Dershowitz, N. (1989). Completion for rewriting modulo a congruence. *Theoretical Computer Science*, 67(2-3) :173–201. Elsevier Science Publishers B. V.

[Balland et al., 2007] Balland, E., Brauner, P., Kopetz, R., Moreau, P.-E., and Reilles, A. (2007). Tom : Piggybacking rewriting on java. In Baader, F., editor, *18th International Conference on Term Rewriting and Applications - RTA 2007*, volume 4533 of *Lecture Notes in Computer Science*, pages 36–47. Springer-Verlag.

[Bandara et al., 2006] Bandara, A., Kakas, A., Lupu, E., and Russo, A. (2006). Using argumentation logic for firewall policy specification and analysis. In *International Workshop on Distributed Systems : Operations and Management Large Scale Management*, volume 4269 of *Lecture Notes in Computer Science*, pages 185–196. Springer-Verlag.

[Barker, 2009] Barker, S. (2009). The next 700 access control models or a unifying meta-model ? In *ACM Symposium on Access control models and technologies*, pages 187–196. ACM.

[Barker et al., 2009] Barker, S., Bertolissi, C., and Fernández, M. (2009). Action control by term rewriting. In *Proceedings of the Third International Workshop on Security and Rewriting Techniques*, volume 234 of *Electronic Notes in Theoretical Computer Science*, pages 19–36. Elsevier Science Publishers B. V.

[Barker and Fernández, 2006] Barker, S. and Fernández, M. (2006). Term rewriting for access control. In *Data and applications security XX*, volume 4127 of *Lecture Notes in Computer Science*, pages 179–193. Springer-Verlag.

[Barker and Fernàndez, 2007] Barker, S. and Fernàndez, M. (2007). Action-status access control as term rewriting. In Nesi, M. and Treinen, R., editors, *Proceedings of the International Workshop on Security and Rewriting Techniques*, pages 3–16.

[Barker and Stuckey, 2003] Barker, S. and Stuckey, P. (2003). Flexible access control policy specification with constraint logic programming. *ACM Transactions on Information and System Security*, 6(4) :501–546. ACM.

[Becker, 2009] Becker, M. (2009). Specification and analysis of dynamic authorisation policies. In *22nd IEEE Computer Security Foundations Symposium*, pages 203–217. IEEE Computer Society.

[Becker and Nanz, 2010] Becker, M. and Nanz, S. (2010). A logic for state-modifying authorization policies. *ACM Transactions on Information and System Security*, 13(3) :1–28. ACM.

[Becker and Sewell, 2004] Becker, M. and Sewell, P. (2004). Cassandra : Flexible trust management, applied to electronic health records. In *17th IEEE Computer Security Foundations Workshop*, pages 139–154. IEEE Computer Society.

[Bell and LaPadula, 1973] Bell, D. and LaPadula, L. (1973). Secure computer systems : Mathematical foundations and model. Technical Report 2547, MITRE.

[Benelbahri and Bouhoula, 2007] Benelbahri, A. and Bouhoula, A. (2007). Tuple based approach for anomalies detection within firewall filtering rules. In *IEEE Symposium on Computers and Communications*, pages 63–70. IEEE Computer Society.

[Bertino et al., 1998] Bertino, E., Bettini, C., Ferrari, E., and Samarati, P. (1998). An access control model supporting periodicity constraints and temporal reasoning. *ACM Transactions on Database Systems*, 23(3) :231–285. ACM.

[Bertino et al., 2003] Bertino, E., Catania, B., Ferrari, E., and Perlasca, P. (2003). A logical framework for reasoning about access control models. *ACM Transactions on Information and System Security*, 6(1) :71–127. ACM.

[Bertolissi and Fernández, 2008] Bertolissi, C. and Fernández, M. (2008). A rewriting framework for the composition of access control policies. In *International ACM SIGPLAN conference on Principles and Practice of Declarative Programming*, pages 217–225. ACM.

[Bertolissi and Fernandez, 2009a] Bertolissi, C. and Fernandez, M. (2009a). An algebraic-functional framework for distributed access control. In *Risks and Security of Internet and Systems*, pages 1–8. IEEE Computer Society.

[Bertolissi and Fernandez, 2009b] Bertolissi, C. and Fernandez, M. (2009b). Distributed event based access control. *International Journal of Information and Computer Security*, 3(3/4) :306–320. Inderscience Publishers.

[Bertolissi et al., 2007] Bertolissi, C., Fernández, M., and Barker, S. (2007). Dynamic event-based access control as term rewriting. In *Data and Applications Security*, volume 4602 of *Lecture Notes in Computer Science*, pages 195–210. Springer-Verlag.

[Bishop, 1991] Bishop, M. (1991). Password Management. In *Compcon Spring'91. Digest of Papers.*, pages 167–169. IEEE Computer Society.

[Bishop, 2002] Bishop, M. (2002). *Computer Security : Art and Science*. Addison-Wesley Professional.

[Bouajjani et al., 2006] Bouajjani, A., Habermehl, P., Rogalewicz, A., and Vojnar, T. (2006). Abstract regular tree model checking. In *7th International Workshop on Verification of Infinite-State Systems*, volume 149 of *Electronic Notes in Theoretical Computer Science*, pages 37–48. Elsevier Science Publishers B. V.

[Boudet and Comon, 1995] Boudet, A. and Comon, H. (1995). Résolution de contraintes symboliques. Notes du cours de DEA d'informatique de Paris 11.

[Bouhoula, 1996] Bouhoula, A. (1996). Using induction and rewriting to verify and complete parameterized specifications. *Theoretical Computer Science*, 170(1-2) :245–276. Elsevier Science Publishers B. V.

[Bouhoula, 1998] Bouhoula, A. (1998). Mécanisation du raisonnement par récurrence. Habilitation à diriger des recherches. Université Henri Poincaré - Nancy 1.

[Bouhoula, 2009] Bouhoula, A. (2009). Simultaneous checking of completeness and ground confluence for algebraic specifications. *ACM Transactions on Computational Logic*, 10(3) :20. ACM.

[Bouhoula and Rusinowitch, 1995] Bouhoula, A. and Rusinowitch, M. (1995). Spike : A system for automatic inductive proofs. In *4th International Conference on Algebraic Methodology and Software Technology*, volume 936 of *Lecture Notes in Computer Science*, pages 576–577. Springer-Verlag.

[Bourdier, 2011a] Bourdier, T. (2011a). Specification, analysis and transformation of security policies via rewriting techniques. *Journal of Information Assurance and Security*, 6(5) :357–368. Dynamic Publishers Inc, USA.

[Bourdier, 2011b] Bourdier, T. (2011b). Tree automata based semantics of firewalls. In *6th International Conference on Network Architectures and Information Systems Security*, pages 171–178. IEEE Computer Society.

[Bourdier and Cirstea, 2010] Bourdier, T. and Cirstea, H. (2010). Constrained rewriting in regular theories. INRIA Research Report (http://hal.inria.fr/inria-00456848/en/).

[Bourdier and Cirstea, 2011] Bourdier, T. and Cirstea, H. (2011). Symbolic analysis of network security policies using rewrite systems. In *13th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming*. ACM.

[Bourdier et al., 2010a] Bourdier, T., Cirstea, H., Dougherty, D., and Kirchner, H. (2010a). Extensional and intensional strategies. In *9th International Workshop on Reduction Strategies in Rewriting and Programming*, volume 15 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–19. Open Publishing Association.

[Bourdier et al., 2010b] Bourdier, T., Cirstea, H., Jaume, M., and Kirchner, H. (2010b). On formal specification and analysis of security policies. Available as an INRIA Research Report (http://hal.inria.fr/inria-00429240/). Grande Region Security and Reliability Day.

[Bourdier et al., 2010c] Bourdier, T., Cirstea, H., Jaume, M., and Kirchner, H. (2010c). Rule-based specification and analysis of security policies. Available as

an INRIA Research Report (http ://hal.inria.fr/inria-00552221/). 5th International Workshop on Security and Rewriting Techniques.

[Bourdier et al., 2011] Bourdier, T., Cirstea, H., Jaume, M., and Kirchner, H. (2011). Formal specification and validation of security policies. In *Foundations & Practice of Security*, volume 6888 of *Lecture Notes in Computer Science*, page to appear. Springer-Verlag.

[Bourdier et al., 2009] Bourdier, T., Cirstea, H., Moreau, P.-E., and de Oliveira, A. S. (2009). Analysis of lattice-based access control policies using rewriting systems and tom. In *Proceedings of the 1st Luxembourg Day on Security and Reliability*, pages 33–40. ISBN 2-919940-84-8.

[Carrasco et al., 2007] Carrasco, R. C., Daciuk, J., and Forcada, M. L. (2007). An implementation of deterministic tree automata minimization. In *12th International Conference on Implementation and Application of Automata*, volume 4783 of *Lecture Notes in Computer Science*, pages 122–129. Springer-Verlag.

[Cirstea et al., 2005] Cirstea, H., Moreau, P., and Reilles, A. (2005). Rule-based Programming in Java For Protocol Verification. In *5th International Workshop on Rewriting Logic and Its Applications*, volume 117 of *Electronic Notes in Theoretical Computer Science*, pages 209–227. Elsevier Science Publishers B. V.

[Cirstea et al., 2009] Cirstea, H., Moreau, P.-E., and Santana de Oliveira, A. (2009). Rewrite based specification of access control policies. In *3rd International Workshop on Security and Rewriting Techniques*, volume 234 of *Electronic Notes in Theoretical Computer Science*, pages 37–54. Elsevier Science Publishers B. V.

[Codd, 1972] Codd, E. F. (1972). *Relational Completeness of Data Base Sublanguages*, pages 65–98. Prentice Hall Inc.

[Cohn, 1981] Cohn, P. M. (1981). *Universal algebra*, volume 6 of *Mathematics and its applications*. D. Reidel Publishing Company. Previous ed. : New York : Harper & Row, 1965.

[Commission of the European Communities, 1991] Commission of the European Communities (1991). Information technology security evaluation criteria (itsec) – provisional harmonized criteria. Office for Official Publications of the European Communities.

[Comon, 1992] Comon, H. (1992). Résolution de contraintes dans les algèbres de termes. Habilitation à diriger des recherches. Université de Paris Sud.

[Comon, 1998a] Comon, H. (1998a). Completion of rewrite systems with membership constraints. Part I : deduction rules. *Journal of Symbolic Computation*, 25 :397–419. Elsevier Science Publishers B. V.

[Comon, 1998b] Comon, H. (1998b). Completion of rewrite systems with membership constraints. Part II : Constraint solving. *Journal of Symbolic Computation*, 25 :421–453. Elsevier Science Publishers B. V.

[Comon, 2001] Comon, H. (2001). Inductionless induction. In Robinson, A. and Voronkov, A., editors, *Handbook of Automated Reasoning, volume 1*, chapter 14, pages 913–962. Elsevier Science Publishers B. V.

[Comon et al., 2008] Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., and Tommasi, M. (2008). Tree automata techniques and applications. Available at : http://gforge.inria.fr/projects/tata/.

[Comon and Delor, 1994] Comon, H. and Delor, C. (1994). Equational formulae with membership constraints. *Information and Computation*, 112(2) :167–216. Elsevier Science Publishers B. V.

[Comon and Jouannaud, 2003] Comon, H. and Jouannaud, J. (2003). Les termes en logique et en programmation. Available at : www.lix.polytechnique.fr/~jouannaud/articles/cours-tlpo.pdf. Notes de cours - Université de Paris Sud.

[Comon and Lescanne, 1989] Comon, H. and Lescanne, P. (1989). Equational problems and disunification. *Journal of Symbolic Computation*, 7 :371–425. Elsevier Science Publishers B. V.

[Conoboy and Fictner, 2002] Conoboy, B. and Fictner, E. (2002). IP Filter Based Firewalls HOWTO. Available at : http://www.obfuscation.org/ipf/ipf-howto.pdf.

[Contejean et al., 2010] Contejean, É., Paskevich, A., Urbain, X., Courtieu, P., Pons, O., and Forest, J. (2010). A3PAT, an approach for certified automated termination proofs. In *ACM SIGPLAN Workshop on Partial evaluation and program manipulation*, pages 63–72. ACM.

[Courcelle, 1989] Courcelle, B. (1989). *Resolution of Equations in Algebraic Structures*, chapter On recognizable Sets and Tree Automata. Academic Press Inc.

[Cuppens, 2000] Cuppens, F. (2000). Modélisation formelle de la sécurité des systèmes d'informations. Habilitation à diriger des recherches. Université Paul Sabatier.

[Cuppens et al., 2005] Cuppens, F., Cuppens-Boulahia, N., and Garcia-Alfaro, J. (2005). Detection and removal of firewall misconfiguration. In *International Conference on Communication, Network and Information Security*, pages 154–162. ACTA Press.

[Cuppens et al., 2006] Cuppens, F., Cuppens-Boulahia, N., and Garcia Alfaro, J. (2006). Detection of network security component misconfiguration by rewriting and correlation. In *Proceedings of the 5th Conference on Security and Network Architectures joint with the 3rd Conference on Security in Information Systems*.

[Cuppens et al., 2004] Cuppens, F., Cuppens-Boulahia, N., Sans, T., and Miège, A. (2004). A formal approach to specify and deploy a network security policy. In *2nd IFIP TC1 WG1.7 Workshop on Formal Aspects in Security and Trust*, volume 173 of *IFIP Advances in Information and Communication Technology*, pages 203–218. Springer-Verlag.

[Damianou et al., 2002] Damianou, N. C., Bandara, A. K., Sloman, M. S., and Lupu, E. C. (2002). A survey of policy specification approaches. Technical report, Department of Computing, Imperial College of Science Technology and Medicine, London, UK. http://www.doc.ic.ac.uk/~mss/Papers/PolicySurvey.pdf.

[Darmaillacq, 2005] Darmaillacq, V. (2005). Comparaison de divers formalismes pour la modélisation et le test de politiques de sécurité. Projet POTESTAT (POlitiques de sécurité : TEST et Analyse par le Test de systèmes en réseau ouvert).

[Dauchet and Tison, 1990] Dauchet, M. and Tison, S. (1990). The theory of ground rewrite systems is decidable. In *5th Annual Symposium on Logic in Computer Science*, pages 242–248. IEEE Computer Society.

[Dausque, 2005] Dausque, N. (2005). PSSI « Politiques de Sécurité des Systèmes d'Information » & CAPSEC « Comment Adapter une politique de Sécurité pour les Entité du CNRS ». *Sécurité Informatique*, 52 :1–5. Bulletin du Centre National de la Recherche Scientifique.

[Dershowitz et al., 1992] Dershowitz, N., Mitra, S., and Sivakumar, G. (1992). Decidable matching for convergent systems. In *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Computer Science*, pages 589–602. Springer-Verlag.

[Deswarte and Mé, 2002] Deswarte, Y. and Mé, L. (2002). *Sécurité des réseaux et systèmes répartis*. Hermès Science, Lavoisier.

[Dougherty et al., 2007] Dougherty, D., Kirchner, C., Kirchner, H., and Santana de Oliveira, A. (2007). Modular Access Control via Strategic Rewriting. In *12th European Symposium On Research In Computer Security*, volume 4734 of *Lecture Notes in Computer Science*, pages 578–593. Springer-Verlag.

[Dougherty et al., 2006] Dougherty, D. J., Fisler, K., and Krishnamurthi, S. (2006). Specifying and reasoning about dynamic access-control policies. In *3rd International Joint Conference on Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 632–646. Springer-Verlag.

[Durán et al., 2008] Durán, F., Lucas, S., and Meseguer, J. (2008). MTT : The Maude termination tool (system description). In *4th International Joint Conference on Automated Reasoning*, number 5195 in Lecture Notes in Computer Science, pages 313–319. Springer-Verlag.

[Durand, 2005] Durand, I. (2005). Autowrite : A tool for term rewrite systems and tree automata. In *5th International Workshop on Rule-Based Programming*, volume 124 of *Electronic Notes in Theoretical Computer Science*, pages 29–49. Elsevier Science Publishers B. V.

[Echahed and Prost, 2005] Echahed, R. and Prost, F. (2005). Security policy in a declarative style. In *International ACM SIGPLAN conference on Principles and Practice of Declarative Programming*, pages 153–163. ACM.

[Edjlali et al., 1999] Edjlali, G., Acharya, A., and Chaudhary, V. (1999). History-based access control for mobile code. In *Secure Internet Programming*, volume 1603/1999 of *Lecture Notes in Computer Science*, pages 413–431. Springer-Verlag.

[Eronen and Zitting, 2001] Eronen, P. and Zitting, J. (2001). An expert system for analyzing firewall rules. In *Proc. 6th Nordic Worksh. Secure IT Systems*, number IMM-TR-2001-14 in Technical report, pages 100–107. Technical Univ. of Denmark.

[Escobar et al., 2007] Escobar, S., Meseguer, J., and Thati, P. (2007). Narrowing and rewriting logic : from foundations to applications. In *15th Workshop on Func-*

tional and (Constraint) Logic Programming, volume 177 of *Electronic Notes in Theoretical Computer Science*, pages 5–33. Elsevier Science Publishers B. V.

[Falke and Kapur, 2006] Falke, S. and Kapur, D. (2006). Inductive decidability using implicit induction. In *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 4246 of *Lecture Notes in Computer Science*, pages 45–59. Springer-Verlag.

[Fernandes, 2001] Fernandes, T. (2001). *Les politiques de sécurité*. PhD thesis, Groupe de recherche LSFM, département d'informatique, Université Laval, Canada.

[Fernandez, 1992] Fernandez, M. (1992). Narrowing based procedures for equational disunification. *Applicable Algebra in Engineering, Communication and Computing*, 3 :1–26. Springer-Verlag.

[Feuillade et al., 2004] Feuillade, G., Genet, T., and Viet Triem Tong, V. (2004). Reachability Analysis over Term Rewriting Systems. *Journal of Automated Reasoning*, 33 (3-4) :341–383. Springer-Verlag.

[Fruhwirth et al., 1991] Fruhwirth, T., Shapiro, E., Vardi, M., and Yardeni, E. (1991). Logic programs as types for logic programs. In *6th Annual Symposium on Logic in Computer Science*, pages 300–309. IEEE Computer Society.

[Fuller and Li, 2006] Fuller, V. and Li, T. (2006). *Classless Inter-Domain Routing (CIDR) : The Internet Address Assignment and Aggregation Plan*. The Internet Society. RFC 4632.

[Gécseg and Steinby, 1997] Gécseg, F. and Steinby, M. (1997). *Handbook of formal languages*, volume 3 : Beyond words, chapter Tree languages. Springer-Verlag.

[Gelfond and Lifschitz, 1998] Gelfond, M. and Lifschitz, V. (1998). Action languages. *Electronic Transactions on Artificial Intelligence*, 2 :193–210. Linköping University Electronic Press.

[Genet and Klay, 2000] Genet, T. and Klay, F. (2000). Rewriting for cryptographic protocol verification. In *17th International Conference on Automated Deduction*, volume 1831 of *Lecture Notes in Computer Science*, pages 271–290. Springer-Verlag.

[Giesl et al., 2006] Giesl, J., Schneider-Kamp, P., and Thiemann, R. (2006). AProVE 1.2 : Automatic termination proofs in the dependency pair framework. In *3rd International Joint Conference on Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 281–286. Springer-Verlag.

[Gligor, 1995] Gligor, V. (1995). Characteristics of role-based access control. In *Proceedings of the 1st ACM Workshop on Role-based access control*. ACM.

[Goubault-Larrecq, 2005] Goubault-Larrecq, J. (2005). Deciding H1 by resolution. *Information Processing Letters*, 95(3) :401–408. Elsevier Science Publishers B. V.

[Gouda and Liu, 2004] Gouda, M. and Liu, A. (2004). Firewall design : Consistency, completeness, and compactness. In *24th IEEE International Conference on Distributed Computing Systems*, pages 320–327. IEEE Computer Society.

[Graham and Denning, 1972] Graham, G. and Denning, P. (1972). Protection : principles and practice. In *the American Federation of Information Processing Societies Fall Joint Computer Conference*, pages 417–429. AFIPS Press.

[Greco et al., 1992] Greco, S., Leone, N., and Rullo, P. (1992). Complex : an object-oriented logic programming system. *IEEE Transactions on Knowledge and Data Engineering*, 4(4) :344–359. IEEE Computer Society.

[Habib et al., 2009] Habib, L., Jaume, M., and Morisset, C. (2009). Formal definition and comparison of access control models. *Journal of Information Assurance and Security*, 4(4) :372–381. Dynamic Publishers Inc, USA.

[Halpern and Weissman, 2008] Halpern, J. and Weissman, V. (2008). Using first-order logic to reason about policies. *ACM Transactions on Information and System Security*, 11(4) :1–41. ACM.

[Hamdi et al., 2007] Hamdi, H., Mosbah, M., and Bouhoula, A. (2007). A domain specific language for securing distributed systems. In *2nd International Conference on Systems and Networks Communications*. IEEE Computer Society.

[Hamed and Al-Shaer, 2006] Hamed, H. and Al-Shaer, E. (2006). Taxonomy of conflicts in network security policies. *IEEE Communications Magazine*, 44(3) :134–141. IEEE Computer Society.

[Hamed et al., 2005] Hamed, H., Al-Shaer, E., and Marrero, W. (2005). Modeling and verification of IPSec and VPN security policies. In *13th IEEE International Conference on Network Protocols*, pages 259–278. IEEE Computer Society.

[Hardin et al., 2006] Hardin, T., Jaume, M., and Morisset, C. (2006). Access control and rewrite systems. In *Proceedings of the 1st International Workshop on Security and Rewriting Techniques*.

[Hazelhurst, 2000] Hazelhurst, S. (2000). Algorithms for analysing firewall and router access lists. Technical Report TR-WITS-CS-1999-5, University of the Witwatersrand, South Africa.

[Heintze and Riecke, 1998] Heintze, N. and Riecke, J. (1998). The SLam calculus : programming with secrecy and integrity. In *ACM SIGPLAN-SIGACT Symposium on Principles of programming languages*, pages 365–377. ACM.

[Hirokawa and Middeldorp, 2005] Hirokawa, N. and Middeldorp, A. (2005). Tyrolean termination tool : Techniques and features. *Information and Computation*, 205(4) :474–511. Elsevier Science Publishers B. V.

[Hoot, 1992] Hoot, C. (1992). Completion for constrained term rewriting systems. In *3rd International Workshop on Conditional Term Rewriting Systems*, volume 656 of *Lecture Notes in Computer Science*, pages 408–423. Springer-Verlag.

[Huet and Hullot, 1982] Huet, G. and Hullot, J. (1982). Proofs by induction in equational theories with constructors. *Journal of Computer and System Sciences*, 25(2) :239–266. Elsevier Science Publishers B. V.

[ISO, 2009] ISO (2009). Common Criteria for Information Technology Security Evaluation. ISO/CEI 15408.

[Jacquemard, 1996a] Jacquemard, F. (1996a). *Automates d'arbres et réécriture de termes*. PhD thesis, Université de Paris-Sud, UFR scientifique d'Orsay.

[Jacquemard, 1996b] Jacquemard, F. (1996b). Decidable approximations of term rewriting systems. In *7th International Conference on Rewriting Techniques and*

Applications, volume 1103 of *Lecture Notes in Computer Science*, pages 362–376. Springer-Verlag.

[Jajodia et al., 2001] Jajodia, S., Samarati, P., Sapino, M., and Subrahmanian, V. (2001). Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(2) :214–260. ACM.

[Jajodia et al., 1997] Jajodia, S., Samarati, P., and Subrahmanian, V. S. (1997). A logical language for expressing authorizations. In *IEEE Symposium on Security and Privacy*, pages 31–42. IEEE Computer Society.

[Jaume, 2010] Jaume, M. (2010). Security rules versus security properties. In *6th International Conference on Information Systems Security*, volume 6503 of *Lecture Notes in Computer Science*, pages 231–245. Springer-Verlag.

[Jaume and Morisset, 2005] Jaume, M. and Morisset, C. (2005). Formalisation and implementation of access control models. In *International Symposium on Information Technology : Coding and Computing*, pages 703–708. IEEE Computer Society.

[Jaume and Morisset, 2006] Jaume, M. and Morisset, C. (2006). Towards a formal specification of access control. In *Proceedings of the Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis*, pages 213–232.

[Jouannaud and Kirchner, 1991] Jouannaud, J.-P. and Kirchner, C. (1991). Solving equations in abstract algebras : a rule-based survey of unification. In *Computational Logic : Essays in Honor of Alan Robinson*, chapter 8, pages 257–321. The MIT-Press.

[Jouannaud and Kirchner, 1986] Jouannaud, J.-P. and Kirchner, H. (1986). Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing*, 15(4) :1155–1194. Society for Industrial and Applied Mathematics.

[Kalam et al., 2003] Kalam, A., Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C., and Trouessin, G. (2003). Organization based access control. In *IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 120–131. IEEE Computer Society.

[Kamara et al., 2003] Kamara, S., Fahmy, S., Schultz, E., Kerschbaum, F., and Frantzen, M. (2003). Analysis of vulnerabilities in internet firewalls. *Computers & Security*, 22 :214–232. Elsevier Science Publishers B. V.

[Kirchner et al., 2008] Kirchner, C., Kirchner, F., and Kirchner, H. (2008). Strategic computations and deductions. In Benzmueller, C., Brown, C., Siekman, J., and Statman, R., editors, *Reasoning in Simple Type Theory – Festschrift in Honor of Peter B. Andrews on his 70th Birthday*, pages 339–364. College Publications.

[Kirchner and Kirchner, 1989] Kirchner, C. and Kirchner, H. (1989). Constrained equational reasoning. In *International Symposium on Symbolic and Algebraic Computation*, pages 382–389. ACM.

[Kirchner and Kirchner, 2006] Kirchner, C. and Kirchner, H. (2006). Rewriting Solving Proving. Preliminary version of a book. Available at : http://www.loria.fr/~hkirchne/rsp.pdf.

[Kirchner et al., 1990] Kirchner, C., Kirchner, H., and Rusinowitch, M. (1990). Deduction with symbolic contraints. *Revue d'intelligence artificielle*, 4(3) :9–52. Hermès Science, Lavoisier.

[Kirchner et al., 2009] Kirchner, C., Kirchner, H., and Santana de Oliveira, A. (2009). Analysis of rewrite-based access control policies. In *3rd International Workshop on Security and Rewriting Techniques*, volume 234 of *Electronic Notes in Theoretical Computer Science*, pages 55–75. Elsevier Science Publishers B. V.

[Lampson, 1971] Lampson, B. (1971). Protection. In *5th Princeton Conf. on Information Sciences and Systems*, pages 18–24. Princeton University.

[Lengál, 2010] Lengál, O. (2010). An efficient tree automata library. Master's thesis, Brno university of technology.

[Limet and Réty, 1998] Limet, S. and Réty, P. (1998). Solving disequations modulo some class of rewrite systems. In *9th International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 121–135. Springer-Verlag.

[Liu, 2008] Liu, A. (2008). Formal verification of firewall policies. In *IEEE International Conference on Communications*, pages 1494 – 1498. IEEE Computer Society.

[Liu and Gouda, 2005] Liu, A. and Gouda, M. (2005). Complete redundancy detection in firewalls. In *Data and Applications Security XIX*, volume 3654 of *Lecture Notes in Computer Science*, pages 193–206. Springer-Verlag.

[Liu et al., 2005] Liu, A., Gouda, M., Ma, H., and Ngu, A. (2005). Firewall queries. In *8th International Conference on Principles of Distributed Systems*, volume 3544 of *Lecture Notes in Computer Science*, pages 197–212. Springer-Verlag.

[Liu and Gouda, 2009] Liu, A. X. and Gouda, M. G. (2009). Firewall policy queries. *IEEE Transactions on Parallel and Distributed Systems*, 20(6) :766–777. IEEE Computer Society.

[Marchand, 2003] Marchand, P. (2003). *Mathématiques Discrètes - Automates, langages, logique et décidabilité*. Dunod.

[Marmorstein and Kearns, 2005] Marmorstein, R. and Kearns, P. (2005). An open source solution for testing NAT'd and nested iptables firewalls. In *19th Conference on Systems Administration*, pages 103–112, Berkeley, CA, USA. USENIX Association.

[Masson et al., 2010] Masson, P.-A., Potet, M.-L., Julliand, J., Tissot, R., Debois, G., Legeard, B., Chetali, B., Bouquet, F., Jaffuel, E., Van Aertrick, L., Andronick, J., and Haddad, A. (2010). An access control model based testing approach for smart card applications : Results of the POSE project. *Journal of Information Assurance and Security*, 5(1) :335–351. Dynamic Publishers Inc, USA.

[McDermott et al., 1998] McDermott, D. et al. (1998). PDDL – the Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control. CVC TR-98-003/DCS TR-1165.

[McLean, 1988] McLean, J. (1988). The algebra of security. In *IEEE Symposium on Security and Privacy*, pages 2–7. IEEE Computer Society.

[Mé and Deswarte, 2006] Mé, L. and Deswarte, Y. (2006). *Sécurité des systèmes d'information*. Hermès Science, Lavoisier.

[Monin, 2000] Monin, J. (2000). *Introduction aux méthodes formelles*. Hermès Science, Lavoisier.

[Morisset, 2007] Morisset, C. (2007). *Sémantique des systèmes de contrôle d'accès – Définition d'un cadre sémantique pour la spécification, l'implantation et la comparaison de modèles de contrôle d'accès*. PhD thesis, Université Pierre et Marie Curie.

[Nelson et al., 2010] Nelson, T., Barratt, C., Dougherty, D., Fisler, K., and Krishnamurthi, S. (2010). The margrave tool for firewall analysis. In *International Conference on Large Installation System Administration*, pages 1–8. USENIX Association.

[Nieslon et al., 2002] Nieslon, F., Nielson, H., and H., S. (2002). Normalizable horn clauses, strongly recognizable relations and spi. In *Static Analysis Symposium*, volume 2477 of *Lecture Notes in Computer Science*, pages 20–35. Springer-Verlag.

[Northern Illinois University, 2010] Northern Illinois University (2010). Northern illinois university electronic mail (e-mail) policy. http://www.its.niu.edu/its/Policies/email_pol.shtml.

[Peltier, 2009] Peltier, N. (2009). Constructing infinite models represented by tree automata. *Annals of Mathematics and Artificial Intelligence*, 56(1) :65–85. Springer-Verlag.

[Rocha, 2011] Rocha, C. (2011). The maude invariant analyzer tool. http://www.camilorocha.info/software/inva.

[Rocha and Meseguer, 2011] Rocha, C. and Meseguer, J. (2011). Proving safety properties of rewrite theories. In *4th Conference on Algebra and Coalgebra in Computer Science*, volume 6859 of *Lecture Notes in Computer Science*, pages 314–328. Springer-Verlag.

[Russell, 2002] Russell, R. (2002). Linux 2.4 packet filtering howto. Available at : http://www.netfilter.org/documentation.

[Russo, 1966] Russo, S. (1966). L'information, source de pouvoir. *Économie rurale*, 69 :3–10. Société Française d'Économie Rurale.

[Rusu, 2010] Rusu, V. (2010). Combining theorem proving and narrowing for rewriting-logic specifications. In *Tests and Proofs*, volume 6143 of *Lecture Notes in Computer Science*, pages 135–150. Springer-Verlag.

[Samarati and de Vimercati, 2001] Samarati, P. and de Vimercati, S. (2001). Access control : Policies, models, and mechanisms. In *Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*, pages 137–196. Springer-Verlag.

[Sandhu, 1993] Sandhu, R. (1993). Lattice-based access control models. *IEEE Computer*, 26(11) :9–19. IEEE Computer Society.

[Sandhu et al., 1996] Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. (1996). Role-based access control models. *IEEE Computer*, 29 :38–47. IEEE Computer Society.

[Sandhu and Samarati, 1994] Sandhu, R. and Samarati, P. (1994). Access control : Principles and practice. *IEEE Communications*, 32(9) :40–48. IEEE Computer Society.

[Santana de Oliveira, 2007] Santana de Oliveira, A. (2007). Rewriting-based access control policies. In *Proceedings of the 1st International Workshop on Security and Rewriting Techniques*, volume 171 of *Electronic Notes in Theoretical Computer Science*, pages 59–72. Elsevier Science Publishers B. V.

[Santana de Oliveira, 2008] Santana de Oliveira, A. (2008). *Réécriture et modularité pour les politiques de sécurité*. PhD thesis, Université Henri Poincaré, Nancy.

[Santana de Oliveira et al., 2007] Santana de Oliveira, A., Wang, E., Kirchner, C., and Kirchner, H. (2007). Weaving rewrite-based access control policies. In *ACM Workshop on Formal methods in security engineering*, pages 71–80. ACM.

[Sasturkar et al., 2006] Sasturkar, A., Yang, P., Stoller, S., and Ramakrishnan, C. (2006). Policy analysis for administrative role based access control. In *Proceedings of the 19th IEEE workshop on Computer Security Foundations*, pages 124–138. IEEE Computer Society.

[Schneider, 2000] Schneider, F. (2000). Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1) :30–50. ACM.

[Souayeh et al., 2010] Souayeh, N., Bouhoula, A., and Jacquemard, F. (2010). Automatic Validation of Firewall Configurations using SMT Solvers. *Journal of Information Assurance and Security*, 5(1) :561–568. Dynamic Publishers Inc, USA.

[Sperschneider and Antoniou, 1991] Sperschneider, V. and Antoniou, G. (1991). *Logic : A Fundation for Computer Science*. International Computer Science Series. Addison-Wesley Publishing Company.

[Srisuresh and Holdrege, 1999] Srisuresh, P. and Holdrege, M. (1999). *IP Network Address Translator (NAT) Terminology and Considerations*. The Internet Society. RFC 2663.

[Sterling and Shapiro, 1994] Sterling, L. and Shapiro, E. (1994). *The art of Prolog*. MIT press.

[Sterne, 1991] Sterne, D. (1991). On the buzzword "security policy". In *IEEE Symposium on Security and Privacy*, pages 219–231. IEEE Computer Society.

[Stratulat, 2005] Stratulat, S. (2005). Automatic 'Descente Infinie' induction reasoning. In *TABLEAUX*, volume 3702 of *Lecture Notes in Computer Science*, pages 262–276. Springer-Verlag.

[Stratulat, 2011] Stratulat, S. (2011). Spike-prover : an automated theorem prover based on 'Descente Infinie' induction. http://code.google.com/p/spike-prover/.

[Terese, 2003] Terese (2003). *Term Rewriting Systems*. Cambridge University Press.

[Thatcher and Wright, 1968] Thatcher, J. W. and Wright, J. B. (1968). Generalized finite automata theory with an application to a decision problem of second-order logic. *Theory of Computing Systems*, 2(1) :57–81. Springer-Verlag.

[Thépaut, 2002] Thépaut, Y. (2002). *Pouvoir, Information, Économie*. Editions Economica.

[Tison, 2000] Tison, S. (2000). Tree automata and term rewrite systems. In *11th International Conference on Rewriting Techniques and Applications*, volume 1833 of *Lecture Notes in Computer Science*, pages 27–30. Springer-Verlag.

[Ullman, 1988] Ullman, J. (1988). *Principles of database and knowledge-base systems, Vol. I.* Computer Science Press, Inc. New York, NY, USA.

[Uribe and Cheung, 2007] Uribe, T. and Cheung, S. (2007). Automatic analysis of firewall and network intrusion detection system configurations. *Journal of Computer Security*, 15(6) :691–715. IOS Press.

[Wijesekera and Jajodia, 2002] Wijesekera, D. and Jajodia, S. (2002). Policy algebras for access control - the predicate case. In *9th ACM Conference on Computer and Communications Security*, pages 171–180. ACM.

[Woo and Lam, 1993] Woo, T. and Lam, S. (1993). Authorization in distributed systems : A new approach. *Journal of Computer Security*, 2(2-3) :107–136. IOS Press.

[Yuan et al., 2006] Yuan, L., Mai, J., Su, Z., Chen, H., Chuah, C.-N., and Mohapatra, P. (2006). FIREMAN : A toolkit for firewall modeling and analysis. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE Computer Society.

Résumé Concevoir et mettre en œuvre des méthodes pour la spécification, l'analyse et la vérification de logiciels et de systèmes sont les principaux moteurs des activités de recherche présentées dans ce manuscrit. Dans ce cadre, nos travaux se positionnent dans la catégorie dite des méthodes formelles appartenant à la communauté plus large du génie logiciel. À l'interface des travaux théoriques et applicatifs, notre objectif est de contribuer aux méthodes permettant d'assurer la correction et la sûreté des systèmes (fonctionnalité, sécurité, fiabilité, ...) en développant ou en améliorant des langages de spécification, des techniques et des outils permettant leur analyse formelle. Dans ce but, nous nous sommes attaché dans cette thèse à proposer et à étudier un cadre formel permettant la définition de politiques de sécurité et la vérification de leurs propriétés. À cet effet, nous avons proposé un cadre pour la spécification de politiques de sécurité basé sur une approche modulaire dans laquelle une politique est vue comme la composition d'un modèle de sécurité et d'une configuration. Nous avons investigué les possibilités offertes par de telles spécifications lorsque les modèles sont exprimés au moyen de contraintes du premier ordre et les configurations au moyen de programmes logiques. En particulier, nous avons proposé un algorithme permettant de transformer une politique exprimée dans un modèle donné vers une autre politique équivalente (au sens où elle engendre les mêmes autorisations) exprimée dans un autre modèle. Dans un second temps, nous nous sommes proposé de tenir compte des aspects dynamiques de la configuration d'une politique vue comme un état du système sur lequel la politique est mise en oeuvre et où chaque action est associée à une procédure de modification des états. Nous avons proposé un langage formel simple pour spécifier séparément les systèmes et les politiques de sécurité puis avons donné une sémantique des spécifications exprimées dans ce cadre sous la forme de systèmes de réécriture. Nous nous sommes ensuite attachés à montrer que les systèmes de réécriture obtenus permettent l'étude de propriétés de sécurité. Dans une troisième partie, nous nous sommes focalisé sur les mécanismes permettant la mise en oeuvre de politiques de sécurité dans les réseaux. Dans ce cadre, nous avons proposé une spécification des firewalls et de leurs compositions basée sur les automates d'arbres et les systèmes de réécriture puis avons montré en quoi ces spécifications nous permettent d'analyser de façon automatique les politiques de sécurité sous-jacentes.

Mots-clés Politiques de sécurité, méthodes formelles, systèmes de réécriture, spécification, vérification.

Abstract Designing and applying formal methods for specifying, analyzing and verifying softwares and systems are the main driving forces behind the work presented in this manuscript. In this context, our activities fall into the category of formal methods belonging to the wider community of software engineering. At the interface between theoretical and applied research, our aim is to contribute to the methods ensuring the correction and the safety of systems (security, reliability, ...) by developing or by improving specification languages, techniques and tools allowing their formal analysis. In this purpose, we became attached in this thesis to propose and to study a formal framework allowing the specification of security policies and the verification of their properties. We first proposed a framework for specifying security policies based on a modular approach in which policies are seen as a composition of security models and configurations. We investigated the possibilities opened by such specifications when models are expressed by means of first order constraints and configurations by means of logical programs. In particular, we proposed an algorithm allowing the transformation of a security policy expressed in a given model towards another equivalent policy expressed in another model. Secondly, we suggested taking into account dynamic aspects of policy configurations which can be seen as states of the system on which the policy is applied and where each action is associated with a procedure of states modification. We proposed a simple formal language to specify separately systems and security policies and then gave a semantics of specifications expressed in this framework under the form of rewriting systems. We then attempted to show that the obtained rewriting systems allow the analysis of security properties. In the third part, we focused on mechanisms enforcing security policies in networks. In this context, we proposed a specification of firewalls and their compositions based on tree automata and rewriting systems and then showed how these specifications allow us to analyze in an automatic way the underlying security policies.

Keywords Security policies, formal methods, rewriting systems, specification, verification.