Titre: Title:	Spécification et validation de protocoles de sécurité
Auteur: Author:	Stéphane Lafrance
Date:	2005
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lafrance, S. (2005). Spécification et validation de protocoles de sécurité [Ph.D. thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7577/

UNIVERSITÉ DE MONTRÉAL

SPÉCIFICATION ET VALIDATION DE PROTOCOLES DE SÉCURITÉ

STÉPHANE LAFRANCE
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIAE DOCTOR (Ph.D)
(GÉNIE INFORMATIQUE)
AVRIL 2005

Your file Votre référence
ISBN: 978-0-494-16999-5
Our file Notre référence
ISBN: 978-0-494-16999-5

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

SPÉCIFICATION ET VALIDATION DE PROTOCOLES DE SÉCURITÉ

présentée par: LAFRANCE Stéphane
en vue de l'obtention du diplôme de: PhilosophiæDoctor
a été dûment acceptée par le jury d'examen constitué de:

M. FERNANDEZ José, Ph.D., président
M. MULLINS John, Ph.D., membre et directeur de recherche
M. ABOULHAMID El Mostapha, Ph.D., membre
M. ECHAHED Rachid, Doctorat, membre
M. MALHAMÉ Roland, Ph.D., examinateur externe

À Olivier.

REMERCIEMENTS

Je tiens tout d'abord à remercier tous les membres de mon jury, et tout particulièrement mon directeur, John Mullins, pour avoir pris le temps de m'enseigner ce sujet, pour son soutien constant et pour toutes ses anecdotes. C'est grâce à lui que j'ai découvert la recherche en informatique et j'en suis chaleureusement reconnaissant.

Je tient aussi à remercier toute l'équipe du laboratoire CRAC pour ces belles années passées en votre compagnie. Un merci tout spécial à Sardaouna Hamadou, ce cher compagnon de bureau, pour son aide précieux lors de la rédaction et de la correction de mes travaux de recherche.

Je souhaite également à remercier tous mes collègues étudiants - Luc, Maurice, Yassir, Alain et Jonathan - pour les discutions animées sur la terrasse du V.-H. et les parties de bbfoot ; sans votre présence quotidienne mes années d'étude auraient été bien plus maussades.

Je remercie aussi tous les professeurs et étudiants qui ont contribué, de loin ou de près, aux travaux présentés dans cette thèse : Moez Yeddes, Nejib Ben Hadj-Alouane, Fen Lin, Gaétan Hains et Armelle Merlin.

Un gros merci aux Fonds québécois de la recherche sur la nature et les technologies et au Conseil de recherches en sciences naturelles et en génie du Canada pour leur soutien financier durant les première années de mes études doctorales.

Finalement, un remerciement tout spécial à ma famille pour leur support inconditionnel, et tout particulièrement à ma femme, Sol Ah, qui a su m'endurer pendant toutes ses années.

RÉSUMÉ

L'émergence des transactions électroniques via l'Internet a introduit dans notre quotidien de nouvelles préoccupations de sécurité. Afin d'uniformiser ce type de procédure, les règles d'échanges de messages qui constituent une telle transaction sont décrites par un protocole de sécurité. Les protocoles de sécurité peuvent être utilisés pour de nombreux buts : pour échanger des messages confidentiels, pour authentifier des individus, pour se brancher sur un serveur web, etc. Les méthodes cryptographiques nous assurent que tout message confidentiel échangé sur un canal public, lorsqu'il est suffisamment bien crypté, ne pourra être déchiffré par un individus que si celui-ci possède la clé correspondante. Cependant, même lorsque que nous utilisons l'hypothèse de cryptage parfait, le problème de déterminer si un protocole est sécuritaire est très complexe. En effet, de nombreux protocoles se sont révélés non sécuritaires, et dans certains cas plusieurs années après leur introduction.

Dans cette thèse, nous abordons le problème de la vérification de protocoles de sécurité. Plus spécifiquement, nous proposons une méthode générale de spécification et de validation pour cette famille de protocoles, qui inclue les protocoles cryptographiques. Nous présentons une nouvelle algèbre de processus, nommée SPPA, qui permet une spécification explicite des échanges de messages entre les participants d'un protocole et des manipulations cryptographiques accomplies par chacun. Une extension symbolique de cette algèbre de processus est également offerte. Nous introduisons ensuite la propriété de sécurité BNAI qui est une formalisation du concept d'interférence admissible munie d'une méthode de vérification basée sur l'équivalence de bisimulation. Nous démontrons que BNAI satisfait certaines propriétés de compositionalité par rapport aux principaux opérateurs de SPPA. Nous prouvons aussi que BNAI, ainsi que d'autres propriétés de non interférence, ne sont pas définissables dans le μ-calcul. De plus, nous montrons comment utiliser BNAI afin de valider certaines propriétés de sécurité, notamment la confidentialité, l'authentification et la vulnérabilité face aux attaques de déni de service.

ABSTRACT

The growing use of e-commerce over the Internet has introduced new security concerns. In order to standardise such procedure, the rules for exchanging messages within such transaction are commonly described by a security protocol. Security protocols are useful for different purpose such as exchanging secret messages, authenticating other users, connecting on a web server, etc. Cryptographic methods guaranties that any secret message exchanged over a public channel, when well encrypted, will not be read by another user unless he owns the corresponding key. However, even under perfect encryption hypothesis, the problem of deciding whether a protocol is safe is very difficult. Indeed, many security protocols were shown unsafe, and in some cases many years after their introduction.

In this thesis, we investigate the validation problem of security protocols. More specifically, we propose a general method for the specification and validation of these protocols, including cryptographic protocols. We present a new process algebra, named SPPA, which allows explicit specification of messages exchanges between users and cryptographic manipulations achieved by each of them. A symbolic extension of this process algebra is also given. We then introduce the BNAI security property which is an interpretation of the admissible interference concept combined with a verification method based on bisimulation equivalence. We prove that this security property satisfies some compositional properties with respect to SPPA's main operators. We also prove that BNAI, along with others non-interference properties, is not definable in μ-calculus. Moreover, we show how to use BNAI for the verification of specific security properties such as confidentiality, authentication and vulnerability against denial of service.

TABLE DES MATIÈRES

LISTE DES FIGURES

CHAPITRE 1

INTRODUCTION

1.1 Motivations

L'émergence des nouvelles technologies en informatique et en télécommunication a engendré un urgent besoin de méthodes et d'outils de vérification. Ceux-ci ont pour but de s'assurer que les protocoles utilisés lors de communications électroniques soient suffisamment sécuritaires malgré le fait que des échanges d'information sont effectués sur des canaux publics tel l'Internet, donc propices à être interceptés par quiconque. L'élaboration et la validation de politiques de sécurité strictes qui établissent une réglementation sur la conception des protocoles utilisés à ces fins est donc devenue un enjeu majeur qui a capté l'intérêt d'une multitude de chercheurs en informatique.

La validation de protocoles de sécurité est une tâche qui requiert typiquement le développement de méthodes formelles permettant la détection de toute attaque possible sur un protocole donné. Il n'y a pas si longtemps, ces protocoles étaient considérés sécuritaires simplement si personne n'y avait trouvé de faille. Ainsi, certains protocoles furent utilisés pendant plusieurs années avant d'être prouvés non sécuritaires.

1.2 Protocoles de sécurité

Il est possible de classer les différents types de protocoles de sécurité selon les services qu'ils offrent. Par exemple, les protocoles cryptographiques sont caractérisés par l'utilisation du cryptage de données afin d'assurer la confidentialité de certaines données échangées entre les usagers, communément appelés participants. Les services offerts par ces protocoles incluent, notamment, l'authentification des participants et l'échanges de clés et de messages confidentiels ainsi que la création

de connexions fiables pour les flots de données. Par exemple, un protocole d'authentification offre un service d'authentification entre deux participants par l'entremise d'une certaine procédure d'authentification. Selon l'*International Organization for Standardization* (ISO) [63], l'objectif général d'un protocole d'authentification est de

> « *permettre la vérification de l'identité prétendue d'une entité par une autre entité, et l'authenticité de l'entité est uniquement garantie que pour l'instant suivant l'échange d'authentification* ».

L'authentification est donc la pierre angulaire de la sécurité des protocoles de sécurité. Nous nous intéressons également aux protocoles de commerce électronique qui offrent des services reliés, plus spécifiquement, aux transactions électroniques via l'Internet. Parmi ces services, il y a, entre autres, le paiement par carte de crédit et l'émission de reçu.

Afin d'assurer la confidentialité et l'authenticité des messages échangés, les protocoles de sécurité nécessitent régulièrement l'utilisation de stratégies de cryptage, de signature et de hachage. Cependant, malgré l'hypothèse de cryptage parfait (c'est-à-dire qu'un intrus ne peut décrypter un message que s'il possède la clé correspondante), les protocoles de sécurité peuvent tout de même contenir des failles au niveau de leur conception, ce qui les rend vulnérables à des attaques perpétrées par des intrus ayant accès aux réseaux publics sur lesquels les données sont échangées.

1.2.1 Protocole de Needham-Schröder

Un protocole est généralement défini suivant une notation à la *Alice et Bob*, c'est-à-dire par une suite de messages échangés entre deux ou plusieurs participants. Un exemple classique est celui du protocole d'authentification à clé publique de Needham & Schröder [86]. Ce protocole cryptographique utilise le cryptage à clé publique dans le but d'établir une authentification mutuelle entre deux participants (désignés par A et B). Afin de compléter la procédure d'authentification, ce pro-

tocole exige que chaque participant possède la clé publique de son homologue. Ces clés publiques sont respectivement désignées par k_A et k_B. La spécification Alice et Bob du protocole de Needham-Schröder est donnée par les étapes suivantes :

$$\begin{array}{lcc} \text{Message 1 :} & A & \overset{id_A, id_B, \{n_A, id_A\}_{k_B}}{\longrightarrow} \quad B \\ \text{Message 2 :} & B & \overset{id_B, id_A, \{n_A, n_B\}_{k_A}}{\longrightarrow} \quad A \\ \text{Message 3 :} & A & \overset{id_A, id_B, \{n_B\}_{k_B}}{\longrightarrow} \quad B. \end{array}$$

À la première étape du protocole (Message 1), A envoie à B son identificateur (id_A) et celui de B (id_B), en compagnie d'un *nonce* fraîchement généré (n_A) et son identificateur tous deux chiffrés à partir de la clé publique de B (k_B). La notation $\{n_A, id_A\}_{k_B}$ représente le résultat de ce cryptage. Notons aussi qu'un *nonce* désigne un nombre aléatoire. Suite à la réception de cette demande d'authentification, le participant B décrypte le message obtenu et obtient le nonce généré par A. Après vérification que l'identificateur id_A à l'intérieur du message chiffré correspond bien à celui non chiffré, B répond à l'invitation de A en lui retournant le message composé de son identificateur, l'identificateur de A, et le nonce de A et un nouveau nonce (n_B) chiffrés ensemble en utilisant la clé publique de A. Entre la deuxième et la troisième étape du protocole, le participant A doit s'assurer que le nonce chiffré reçu de B correspond bien à celui qu'il a initialement généré. Cette vérification permet à A de s'assurer de l'authenticité de B. Dans l'éventualité d'un succès, A envoie à B le troisième message qui est composé des deux identificateurs ainsi que le nonce de B chiffré avec sa clé publique. Après avoir reçu ce dernier message, le participant B peut authentifier le participant A en vérifiant si le nonce fraîchement reçu correspond à celui qu'il a généré plus tôt.

Bien que le protocole de Needham-Schröder semble sécuritaire au premier coup d'oeil, il en est tout autrement. En effet, Lowe [74] a découvert une faille dans ce protocole pouvant mener à une attaque de type « *man-in-the-middle* ». L'aspect le plus inquiétant à propos de cette attaque, c'est qu'elle fut découverte environ vingt années après l'introduction du protocole... Cette fameuse attaque est seulement réalisable lorsque le participant A initie le protocole d'authentification avec

un participant malveillant E (souvent appelé intrus ou participant ennemi). Dans ce cas, un tel participant malhonnête E peut utiliser l'information obtenue de A afin d'initier une attaque d'authentification avec un tiers participant B, dans laquelle E convainc B qu'il est A. Cette mascarade est réalisable à partir de deux exécutions parallèles du protocole de Needham-Schröder dans lesquels l'intrus E joue un rôle central. Cette attaque est spécifiée dans la notation Alice et Bob de la façon suivante :

Étape 1 :	A	$\overset{id_A,id_E,\{n_A,id_A\}_{k_E}}{\longrightarrow}$	E	(*A initie le protocole avec E*);
Étape 2 :	E	$\overset{id_A,id_B,\{n_A,id_A\}_{k_B}}{\longrightarrow}$	B	(*E initie le protocole avec B*) *en utilisant l'identité de A*);
Étape 3 :	B	$\overset{id_B,id_A,\{n_A,n_B\}_{k_A}}{\longrightarrow}$	E	(*E intercepte la réponse de B*);
Étape 4 :	E	$\overset{id_E,id_A,\{n_A,n_B\}_{k_A}}{\longrightarrow}$	A	(*E transfert la réponse de B vers A*);
Étape 5 :	A	$\overset{id_A,id_E,\{n_B\}_{k_E}}{\longrightarrow}$	E	(*A répond à E*);
Étape 6 :	E	$\overset{id_A,id_B,\{n_B\}_{k_B}}{\longrightarrow}$	B	(*E transfert la réponse de A vers B*).

1.3 Attaques et propriétés de sécurité

La nature des attaques sur un protocole spécifique dépend essentiellement des services offerts par celui-ci. Du coup, les propriétés de sécurité que nous imposons à un protocole, afin de s'assurer qu'il est sécuritaire, dépendent de ses objectifs. Dans ce qui suit, nous exposons brièvement certaines de ces propriétés de sécurité.

1.3.1 Confidentialité

Lors d'une attaque sur un protocole d'échange d'information, un intrus tente de récolter des renseignements à propos du contenu d'un message secret qui ne lui est pas destiné. La *confidentialité* réfère à la propriété de sécurité qui s'assure que toute donnée confidentielle échangée demeure toujours secrète. De toute évidence, le cryptage des messages résout une grande partie des préoccupations de confidentialité, pour autant que l'algorithme de cryptage utilisé soit suffisamment

sécuritaire. Cependant, certaines divulgations de données secrètes peuvent persister. Par exemple, un intrus pourrait exploiter une faille du protocole dans le but d'obtenir les clés privées requises. Si un intrus parvient à lire le contenu intégral d'un message secret, alors il y a clairement violation de la confidentialité. Par contre, devons-nous considérer comme non sécuritaire un protocole qui divulgue seulement une infime partie d'un message secret (par exemple le premier bit d'un numéro de carte de crédit), ou qui divulgue simplement l'existence d'un message secret ? Dans le cadre de cette thèse, nous considérons toute divulgation d'information concernant un message secret comme un non respect de la confidentialité ; ceci inclut toute information, directe ou indirecte, obtenue par l'étude des flots d'information du protocole.

Un système informatique qui est fondé sur une architecture à plusieurs niveaux de sécurité doit inévitablement établir et faire respecter une hiérarchie de confidentialité entre les usagers et les objets. Dans le cas le plus simple où il n'y a que deux niveaux, communément désignés par *haut* (privé) et *bas* (public), nous souhaitons interdire aux usagers et aux programmes du bas niveau d'accéder aux données du haut niveau. Dans ce type d'architecture, le non respect de la confidentialité est généralement illustré par des usagers (ou programmes) de bas niveaux du système qui tentent d'obtenir l'accès à de l'information confidentielle en interagissant simplement avec le système et les autres usagers et en effectuant certaines déductions. D'autre part, une attaque pourrait également provenir d'un usager ou d'un programme du haut niveau qui tente de divulguer des informations confidentielles, auxquelles il a accès, à un complice du bas niveau. Un tel usager ou programme malhonnête du haut niveau est communément appelé Cheval de Troie (*Trojan Horse*). Un Cheval de Troie qui a directement accès à des données confidentielles pourrait exploiter certaines failles du système afin de convoyer des informations secrètes vers une destination de bas niveau. Les règles d'accès MAC (*Mandatory Access Control*) furent introduites afin d'imposer une politique de respect de la confidentialité dans un système composé de plusieurs niveaux de sécurité. Ces règles d'accès empêchent les usagers de lire des objets de plus haut niveau et d'écrire sur

des objets de plus bas niveau (*no read up, no write down*). Cependant, ces règles s'avèrent insuffisantes afin de s'assurer que le système soit totalement confidentiel. En effet, un Cheval de Troie pourrait divulguer de l'information confidentielle tout en respectant les règles MAC. Par exemple, considérons un système composé de deux niveaux de sécurité qui partagent une ressource d'entreposage finie (e.g. un disque dur). Il est alors possible de transmettre de l'information du haut niveau vers le bas niveau de la façon suivante :

> Un Cheval de Troie (du haut niveau) rempli et vide alternativement l'espace d'entreposage que nous supposons aussi de haut niveau. Au même moment, son complice du bas niveau essaye d'écrire sur ce même espace, en décodant chaque échec (c'est-à-dire disque plein) comme un « 0 » et chaque écriture réussie comme un « 1 ».

Cet exemple de canal clandestin binaire permet la transmission d'information confidentielles sans jamais enfreindre les règles d'accès MAC.

Nous sommes donc d'avis que l'élaboration d'une propriété de confidentialité devrait prendre en compte les capacités qu'ont les usagers de bas niveau, plus spécifiquement les intrus, à déduire certaines informations à l'aide de leurs propres observations du système ou du protocole. Comme nous l'avons constaté plus haut, la simple existence d'une corrélation entre des données confidentielles et un comportement observable par un intrus est propice à une divulgation d'information. Cependant, certaines de ces corrélations sont à la fois inévitables et acceptables. Par exemple, considérons un protocole cryptographique dans lequel un participant envoie un message chiffré sur un canal public. Cette situation engendre une déclassification du message puisque que la simple observation du message chiffré sur le canal public permet à l'intrus d'en déduire son existence, sans pour autant en connaître son contenu. Nous sommes d'avis qu'une propriété de confidentialité adéquate doit détecter tout flot d'information allant d'un niveau privé vers un niveau public, à moins que cette déclassification soit accomplie par l'entremise d'un canal spécialement conçu à cet effet.

1.3.2 Authentification

Une deuxième famille d'attaques est composée des attaques sur les protocoles d'authentification. Ces attaques prennent typiquement la forme d'une mascarade : un intrus utilise les données obtenues lors d'une session du protocole ou interceptées sur les canaux publics afin de dérober l'identité d'un des participants du protocole. Dans plusieurs cas, dont le protocole de Needham-Schröder, une attaque est réalisable lorsqu'un premier participant initie le protocole avec un participant malhonnête – l'intrus – qui réutilise l'information transmise afin de lui extirper son identité et ainsi persuader un autre participant qu'il discute avec le premier. Ces attaques sont généralement détectables en observant le comportement du protocole lorsque confronté à un environnement hostile. Lors d'un tel scénario, plusieurs tentatives d'attaques sur le protocole sont initiées et toute attaque victorieuse est rapportée.

1.3.3 Déni de service

La dernière décennie fut témoin de l'émergence des attaques de *déni de service* (DoS), dont le but principal est de priver certains usagers d'accéder à une ressource ou un service spécifique. Une telle attaque met habituellement en évidence les limitations d'un certain participant. Plus spécifiquement, le attaques de DoS prennent typiquement l'une des formes suivantes :

- un intrus inonde un réseau dans le but de bloquer le trafic réseau entre les usagers légitimes ;
- un intrus perturbe la connexion entre deux machines afin de priver l'accès à un service ;
- un intrus empêche un certain usager d'accéder à un service ;
- un intrus perturbe directement le service offert par un système ou un usager.

Une utilisation illégitime des ressources peut également résulter en un DoS. Par exemple, un intrus pourrait utiliser l'emplacement ftp d'un système public afin d'y entreposer des copies piratées d'un logiciel commercial et ainsi consommer une

bonne quantité d'espace disque et créer un impact significatif sur le trafic réseau. D'autre part, certaines attaques de déni de service permettent de perturber des sites Internet sophistiqués à partir d'une quantité limité de ressources. Ainsi, un intrus muni uniquement d'un vieux PC et d'un modem lent pourrait mettre hors d'usage des machines ou des réseaux beaucoup plus puissants.

Ces dernières années, plusieurs sites Internet – principalement à caractère commercial – furent victimes d'attaques de DoS. Parmi celles-ci, on retrouve la célèbre attaque de *SYN flooding* [101] sur le protocole TCP/IP. Depuis 1996, cette attaque par épuisement de ressources fut perpétrée à plusieurs reprises par des intrus ayant la capacité d'initier, avec un minimum d'effort, un grand nombre d'exécutions du protocole avec un même serveur. La possibilité d'utiliser le protocole TCP/IP afin de provoquer ce DoS est partiellement due à la facilité de forger une fausse identité et, conséquemment, la difficulté pour la victime de reconnaître un attaquant. D'autres attaques de DoS furent recensées sur des sites Internet à caractère commercial, dont *Yahoo*, *Ebay* et *E*trade* en février 2000, de même que *Microsoft* en janvier 2001. Dès 1999, une nouvelle forme encore plus dangereuse de DoS, appelée *déni de service distribué* (DDoS), a fait son apparition causant sa part de pagaille. Ces attaques utilisent plusieurs points d'un réseau opérant de concert dans le but d'attaquer un réseau ou un site par l'entremise d'un protocole. Un épuisement de ressources est créé simplement par une quantité accablante de demandes de branchement. Le refoulement de ces demandes forgées, sans pour autant empêcher le trafic légitime, est une tâche très ardue puisque les DDoS sont perpétrés en inondant la victime d'une grande quantité de trafic réseau provenant de différents emplacements et contrôlée par un seul intrus. Quelques outils [34,36,88] furent développés pour l'analyse d'attaques de DDoS effectuées par des applications malicieuses spécifiques, dont *Trin00*, *TFN2K* et *Stacheldraht*.

1.3.4 Sécurité dans les protocoles de commerce électronique

En plus de la confidentialité et de l'authenticité, l'analyse de protocoles de commerce électronique nécessite la définition de propriétés de sécurité spécifiques telles

l'*anonymat*, la *non répudiation*, la *garantie de livraison* et l'*atomicité de l'argent et des biens et services*. L'anonymat des participants est une application particulière de la propriété de confidentialité qui exige que l'identité (nom, courriel, adresse, emplacement, etc.) d'un certain participant à protocole demeure cachée aux autres participants. Notons cependant que comparativement à la confidentialité, l'anonymat est enfreinte seulement lorsqu'une identité est entièrement révélée ; une connaissance partielle d'un nom secret ou d'une adresse confidentielle est non significative si on ne peut pas en déduire leur contenu.

1.3.5 Modélisation des attaques

Une approche intuitive pour découvrir des failles dans un protocole de sécurité consiste à modéliser un intrus spécifique et à observer le comportement du protocole agissant en concurrence avec celui-ci. Par la suite, il suffit de chercher une attaque menée à terme parmi tous les dénouements possibles, et répéter au besoin cette procédure avec d'autres processus ennemis. De toute évidence, cette approche ne convient généralement pas à démontrer qu'un protocole n'admet aucune faille. La complétude d'une telle méthode de validation requiert donc l'interaction du protocole avec un plus puissant attaquant, c'est-à-dire un processus ennemi capable de reproduire toute attaque réalisable par un autre processus ennemi. Malgré le fait que les techniques de modélisation et les algorithmes de vérification diffèrent énormément d'un auteur à l'autre, ce concept presque utopique de plus puissant attaquant est souvent approché par un seul intrus ayant la capacité de communiquer et d'interagir avec le protocole selon les règles suivantes.

- Un intrus peut intercepter tout message envoyé sur un canal public et, par conséquent, acquérir de nouvelles données qui lui permettront de construire de nouveaux messages.
- Un intrus est un participant légitime, et a donc la capacité d'initier le protocole avec tout autre participant ou, réciproquement, de recevoir des invitations de la part des autres participants.
- Un intrus possède certaines connaissances initiales incluant ses clés privées,

toute clé publique accessible aux autres participants, et son propre ensemble de nonces et de messages forgés.

1.4 Aperçu de la thèse

Dans cette thèse, nous présentons une nouvelle méthode de validation pour les protocoles de sécurité. Cette thèse a pour but d'introduire les principales composantes de cette méthode, soit la modélisation des protocoles, l'élaboration de propriétés de sécurités et le développement d'algorithmes permettant la vérification de celles-ci. Bien que nous présentons plusieurs applications pertinentes de notre méthode sur des protocoles connus, l'objectif premier de cette thèse est d'établir les bases théoriques qui ont déjà permis le développement d'un outil de validation automatique de protocoles de sécurité.

Spécification. La première phase de la conception d'une méthode de validation de protocoles de sécurité consiste à déterminer un langage dans lequel nous pouvons exprimer à la fois le protocole et la propriété de sécurité que nous souhaitons faire respecter. D'autre part, la vérification d'une propriété de sécurité nécessite fréquemment que les appels de fonction effectués par les participants – tels le cryptage/décryptage de données et la génération de nombres aléatoires – soient observables. Suivant cet objectif, nous introduisons au Chapitre 3 un nouveau modèle de spécification basé sur une algèbre de processus appelée *Security Protocols Process Algebra* (SPPA). Cette algèbre de processus introduit aussi la notion d'actions de marquage dans le but d'étiqueter tout échange de messages entre les participants du protocole. Nous introduisons également une algèbre de messages et une logique décidable avec lesquelles nous pouvons exprimer des énoncés propres aux protocoles cryptographiques. Au Chapitre 4, nous introduisons une méthode de spécification symbolique capable de manipuler des valeurs symboliques telles les nonces, les clés de session et les adresses forgées. L'idée principale derrière notre approche consiste à assigner à chaque processus SPPA une formule décrivant les valeurs symboliques transportées par sa sémantique. À chacun de ces processus

symboliques, appelés processus contraints, correspond une sémantique opérationnelle symbolique que nous démontrons finie (c'est-à-dire un graphe de transitions avec un nombre fini d'états). D'autre part, nous définissons une équivalence de bisimulation pour les processus contraints qui s'avère être une généralisation de l'équivalence de bisimulation de Milner (pour les processus non symboliques) et pour laquelle nous proposons une technique de preuve cohérente et complète.

Validation. Le point central de cette thèse est l'introduction d'une propriété générique de sécurité appelée *interférence admissible non déterministe par bisimulation* (BNAI). L'idée principale derrière BNAI est de permettre les flots d'information entre différents niveaux de sécurité uniquement à travers un système de déclassification (un système cryptographique par exemple). Au Chapitre 5, nous donnons une caractérisation algébrique de cette propriété qui fournit un algorithme de vérification basé sur une équivalence observationnelle. De plus, nous démontrons que BNAI satisfait certaines propriétés qui mettent en évidence son caractère compositionnel. D'autre part, nous démontrons que BNAI n'est pas définissable dans le μ-calcul modal.

Applications. Des applications non triviales de BNAI pour la vérification de propriétés de sécurité sont illustrées au Chapitre 6. Pour chacune de ces applications, des techniques de preuve cohérentes et complètes sont déduites de la caractérisation algébrique de BNAI. Nous obtenons une propriété de confidentialité qui exige qu'aucun intrus puisse discerner, de façon inadmissible, le comportement normal du protocole du comportement du protocole dans lequel aucune information confidentielle est échangée. Nous obtenons aussi une propriété d'authentification qui vérifie qu'aucun intrus n'a la capacité d'interférer, de façon inadmissible, avec les participants d'un protocole d'authentification. Finalement, nous présentons une propriété de robustesse contre le déni de service qui s'assure que tout comportement coûteux (en terme de ressources CPU ou de ressources mémoire), et qui pourrait causer un épuisement de ressources de la victime, est indépendant de toute tenta-

tive d'attaque. Des illustrations détaillées de ces propriétés sont données au Chapitre 7 à l'aide du protocole *Wide Mouthed Frog*, du protocole d'authentification de Woo-Lam, du protocole d'authentification à clé publique de Needham-Schröder, du protocole de communication TCP/IP et du protocole de paiement électronique sécuritaire 1KP.

CHAPITRE 2

MÉTHODES DE VALIDATION DE PROTOCOLES DE SÉCURITÉ

L'analyse d'un protocole de sécurité commence nécessairement par une phase de modélisation. La spécification rigoureuse d'un protocole de sécurité, dans un objectif de vérification, est un problème notoire [30, 79]. Dans ce chapitre, nous présentons un aperçu des principaux modèles dédiés aux protocoles qui furent proposés au cours des dernières années. Plus spécifiquement, nous présentons deux grandes familles de modèles, soit les modèles de traces et les modèles d'algèbre de processus. Les modèles de traces sont principalement caractérisés par une modélisation des protocoles en termes de règles d'inférence, par opposition à une modélisation en termes d'expressions algébriques et de graphes de transitions pour les algèbres de processus. Nous donnons ensuite un aperçu des principales méthodes de vérification qui ont servi d'inspiration à cette thèse, dont les méthodes de flots d'information, les méthodes symboliques et les méthodes de détection du déni de service. Nous débutons ce chapitre par un bref survol des principales familles d'approches méthodologiques à la validation de protocoles.

2.1 Validation de protocoles

Plusieurs méthodes de validation de protocoles de sécurité, provenant d'une vaste gamme d'approches, sont proposées dans la littérature. Parmi celles-ci, nous pouvons identifier deux grandes familles : le *theorem-proving* et le *model-checking*. Les méthodes de *theorem-proving* consistent à démontrer formellement que le protocole ne possède aucune faille. Ces méthodes offrent une certification incontestable mais elles sont beaucoup plus difficiles à développer d'un point de vue algorithmique. Par conséquence, ces méthodes sont souvent moins propices à une analyse automatisée et à la conception d'un outil de validation.

Les méthodes de *model-checking* nécessitent l'extraction d'un modèle représen-

tant le protocole et la validation, par ce modèle, de la spécification d'une propriété de sécurité. Une propriété de sécurité est typiquement spécifiée par une formule provenant d'une logique compatible avec le modèle formel dans lequel le protocole est spécifié. La logique temporelle CTL [31], la logique TLA [72], la logique modale de Hennessy-Milner [58] et le μ-calcul [87] sont fréquemment utilisés à cette fin.

Plus récemment, les méthodes d'*équivalence-checking* ont démontré leur efficacité pour la vérification de protocoles de sécurité [5,21,32]. Celles-ci se distinguent des méthodes de *model-checking* principalement par leur formalisation des propriétés de sécurité ; elles sont formalisées en terme d'indistinguabilité entre les comportements d'un protocole confronté à divers environnements hostiles. Ainsi, ces méthodes se ramènent généralement à vérifier si le comportement d'un protocole lors d'une attaque équivaut à un comportement correct. Si l'équivalence est satisfaite, alors nous pouvons conclure que le protocole est sécuritaire. Par contre, si le protocole attaqué n'est pas équivalent au protocole sans intrus, alors nous pouvons conclure que l'intrus peut influencer le déroulement du protocole. Évidement, toute méthode d'*équivalence-checking* nécessite un modèle de spécification pour les protocoles et une relation d'équivalence entre ces spécifications afin qu'on puisse comparer leurs comportements.

Les méthodes de validation par typage [1,59,60] constituent une autre importante approche à la validation de protocoles. L'idée générale derrière ces méthodes consiste à formaliser les environnements hostiles à l'aide d'un système de typage dans lequel les protocoles sont représentés par des environnements statiques et les attaques par des types. Ces systèmes de typage offrent des caractérisations algébriques du problème de vérification de protocoles en termes d'ensembles finis de règles de typage (règles d'inférence). La satisfaction d'une propriété de sécurité est généralement une conséquence d'un typage adéquat du protocole.

2.2 Modèles de traces

Les modèles de traces permettent une représentation des protocoles par une suite des règles d'inférence, correspondant aux différentes étapes du protocole. L'ensemble des traces d'un tel modèle est généralement composé des messages que l'on peut déduire à partir d'une configuration initiale et d'applications successives de ces règles d'inférence. Du coup, ces modèles de traces sont souvent restreints à des méthodes de validation basées sur une analyse de l'atteignabilité de messages ou d'états.

Modèle de Dolev-Yao. Dolev & Yao [37] ont proposé un modèle basé sur une spécification des protocole cryptographiques en termes de règles de réécriture et une spécification d'un intrus, appelé saboteur, qui possède un contrôle total sur le mécanisme d'échange de données utilisé par le protocole. Dans ce modèle, la validation d'un protocole est accomplie en s'assurant que le saboteur ne soit jamais en mesure de construire un message correspondant à une attaque. L'identification de ce type de messages dépend évidemment de la propriété de sécurité que nous voulons vérifier. Par exemple, pour la confidentialité, on doit s'assurer que le saboteur ne peut en aucun cas produire un message secret. Pour l'authenticité, on doit s'assurer que le saboteur ne peut pas provoquer une authentification non sollicitée. Des algorithmes de vérification basés sur une analyse des traces du protocole sont développés pour chacune de leurs formalisations de la sécurité.

Grâce à sa simplicité, le modèle de Dolev-Yao a inspiré plusieurs autres méthodes d'analyse de protocoles cryptographiques. Entre autres, Amadio & Lugiez [10] ont introduit une méthode de validation dans laquelle les propriétés de sécurité sont exprimées en terme de problèmes classiques de la théorie de la réécriture, dont le *problème d'atteignabilité*. Le *problème d'atteignabilité du contrôle* dans le modèle de Dolev-Yao fut également étudié par Amadio & Charatonik [9], dans le cas où les participants sont spécifiés en terme de processus récursifs avec des noms générés dynamiquement. Les auteurs présentent une analyse ensembliste,

qui s'avère à la fois conservatrice et décidable, d'une approximation du problème d'atteignabilité par l'entremise d'une sémantique opérationnelle linéaire.

Suivant une approche similaire à celle d'Amadio, Rusinowitch & Turuani [96] ont étudiés la complexité du problème de la sécurité des protocoles dans le cas où il n'y a qu'un nombre fini de sessions. Ils ont démontré que ce problème est NP-complet dans le modèle de Dolev-Yao dès qu'on suppose que la taille des messages n'est pas bornée. De plus, ils offrent une borne linéaire sur la taille des messages nécessaires au saboteur afin de construire une attaque qu'à partir d'un nombre fini de sessions.

2.3 Algèbres de processus

La validation d'une propriété de sécurité est un problème qui va souvent au-delà des problèmes d'atteignabilité, d'où la nécessité d'étendre les modèles de traces, et tout particulièrement le modèle de Dolev-Yao. La vérification des propriétés de non interférence, que nous aborderons à la prochaine section, est un bon exemple de problème non définissable en termes d'atteignabilité. D'autre part, dans le contexte d'une méthode de *model-checking*, la vérification de propriétés de sécurité est habituellement accomplie par l'entremise d'un système de preuve. Ainsi, un bon langage de spécification doit posséder certaines caractérisations algébriques permettant l'élaboration de preuves récursives par rapport à ses structures. (Krishnan [67] et van Glabbeek [107] illustrent de nombreux avantages de la forme récursive des structures syntaxiques des algèbres de processus pour l'analyse de systèmes informatiques).

La théorie de la communication et de la concurrence, telle qu'exposée par Milner [81], prône l'utilisation d'algèbres de processus pour la spécification de systèmes composés de plusieurs participants qui évoluent en concurrence et qui peuvent communiquer entre eux. Plus précisément, un protocole de sécurité peut être représenté par un agglomérat de processus, chacun représentant un participant, qui ont la capacité de communiquer par rendez-vous afin d'échanger des messages. Ainsi, les protocoles sont vus comme des systèmes de transitions dans lesquels les états sont

caractérisés par des affectations de variables et les transitions sont étiquetées par des actions.

Étant donnée une algèbre de processus, nous cherchons à établir une relation d'équivalence entre les processus qui nous permet de déterminer quels processus devraient être considérés sémantiquement équivalents, c'est-à-dire devraient se comporter de façon analogue. En outre, ces relations d'équivalence sont essentielles aux méthodes d'*équivalence-checking*. Parmi ces relation d'équivalences, nous comptons l'*équivalence de traces* qui établit une sémantique par rapport aux langages (traces) des processus, et l'*équivalence de bisimulation* qui établit une sémantique par rapport à leurs comportements.

2.3.1 CCS et CSP

Les algèbres de processus *Communication Concurrency System* (CCS) [81] et *Communicating Sequential Processes* (CSP) [61] sont fondées sur la notion d'action qui permet de modéliser les comportements possibles d'un processus, soit l'émission et la réception d'un message. Elles furent initialement conçues pour la spécification de schémas de communication des composantes de systèmes concurrents qui interagissent par échange de messages. Elles conviennent donc très bien à la description et l'analyse des flots d'information dans un réseau, ainsi qu'à la spécification et la validation de protocoles de sécurité. Leur syntaxe abstraite permet la spécification de processus concurrents à partir d'opérateurs de séquentialité $(\alpha.P)$[1], de somme non déterministe $(P + Q)$, de produit parallèle $(P \parallel P)$, de restriction$(P\backslash L)$ et de dissimulation (P/L)[2]. La sémantique opérationnelle d'un processus CCS ou CSP s'interprète par un graphe de transitions, une extension de la notion d'automate non déterministe dans laquelle nous permettons un nombre infini d'états et nous omettons généralement le concept d'état final. Elle s'obtient d'un ensemble de règles récursives qui décrit le comportement local de chaque opérateur.

L'algèbre de processus *CCS avec passage de paramètres* (*value-passing CCS*)

[1] α est une action.
[2] L un sous-ensemble d'actions.

est une extension de CCS qui offre une formalisation plus précise de la communication entre les agents. Cette extension est obtenue en interprétant, au niveau de la sémantique opérationnelle, un processus avec une variable libre par la famille de tous les processus que l'on peut obtenir en replaçant toute occurrence de la variable libre par un message quelconque. Par exemple, la sémantique du processus à préfixe d'entrée $c(x).P$ correspond à la famille de processus $\{c(a).P[a/x]\}_{a\in\mathcal{M}}$, où $\mathcal{M}$ désigne l'ensemble des messages pouvant être envoyés sur le canal c et $P[a/x]$ désigne le processus dans lequel la variable libre x est substituée par le message a. De toute évidence, cette sémantique peut produire des graphes de transitions infinis : celui correspondant au processus $c(x).P$ ne sera pas à branchement fini dès que l'ensemble $\mathcal{M}$ est infini. Ce problème provoque donc des inconvénients majeurs lors d'une analyse indépendante des participants jouant un rôle de récepteur dans un protocole de sécurité.

2.3.2 Modèle de Lowe

Lowe [74] utilise l'algèbre de processus CSP afin de spécifier, individuellement, chaque participant du protocole d'authentification à clé publique de Needham-Schröder (voir Section 1.2.1). Il définit aussi, toujours en terme de processus CSP, un processus ennemi appelé *intrus le plus général* ou *intrus le plus non déterministe* qui satisfait, grossièrement, les capacités exposées à la Section 1.3.5. Une analyse du produit parallèle des processus CSP modélisant les participants et ce processus ennemi a mené à la découverte de la faille d'authentification exposée à la Section 1.2.1. L'authentification du participant sollicité (respectivement l'authentification du participant qui initie le protocole) est validée en s'assurant que la spécification du protocole combinée à celle de l'intrus est un raffinement, en terme de traces, d'un autre processus CSP dans lequel toute authentification du participant sollicité (resp. toute authentification du participant qui initie le protocole) est précédée par une certaine action qui indique clairement l'intention du participant qui a initié le protocole (resp. l'intention du participant sollicité) pour cette authentification.

Plus récemment, Lowe et al. ont utilisé CSP pour la modélisation formelle et l'analyse des flots d'information dans les systèmes à plusieurs niveaux de sécurité [75], les protocoles d'authentification [26] et les systèmes de détection d'intrusion[3] [92].

2.3.3 Modèle de Schneider

Schneider [98,99] a développé un modèle dans lequel les propriétés de sécurité sont définies à l'aide d'un intrus dont l'identité est connue. Son approche consiste à modéliser le protocole et l'intrus par des processus CSP distincts. Dans ce modèle, le processus ennemi est capable d'altérer le déroulement du protocole via certains canaux non accessibles aux autres participants et un ensemble de connaissances constitué de tout message qu'il peut construire à partir de ses propres clés privées et nonces, ainsi qu'à partir des messages interceptés sur les canaux publics. Cette approche lui permet de considérer diverses tactiques d'attaque, ce qui augmente l'efficacité de sa méthode d'analyse de protocoles par *model-checking*. Schneider illustre sa méthode à l'aide d'une propriété de *confidentialité des messages* :

> tout envoie d'un message secret qui aboutit dans les mains de l'intrus lui était initialement destiné ;

et d'une propriété d'*authentification de message* :

> l'ensemble d'évènements K authentifie l'ensemble L si toute occurrence d'un message provenant de L est précédée d'au moins un message provenant de K.

2.3.4 π-calcul

Le π-calcul est une extension de CCS, introduite par Milner, Parrow & Walker [82], pour la spécification de processus mobiles concurrents. Plus précisément, cette algèbre de processus permet la modélisation de processus réseaux munis d'un

[3] *Intrusion Detection System* (IDS).

mécanisme de communication dynamique. Par exemple, le π-calcul permet la spécification de réseaux nécessitant une gestion dynamique de liens ou de canaux, tel un réseau local dans lequel les usagers doivent effectuer une demande auprès de l'administrateur pour l'obtention d'un canal dynamique ou d'une permission afin d'accéder à une certaine application critique (e.g. une imprimante). Le π-calcul est essentiellement fondé sur la notion de *nom*. Les noms sont utilisés pour désigner à la fois les messages et les canaux, et peuvent être cachés et créés dynamiquement. Ainsi, la syntaxe abstraite du π-calcul permet le passage de noms par les canaux publics, en plus de contenir un opérateur de *restriction de portée* (νnP) et un opérateur de *copie* ($!P$).

La littérature contient plusieurs sémantiques pour le π-calcul et les processus mobiles. Par exemple, Ferrari et al. [51] introduisent un environnement de vérification pour le π-calcul à l'aide d'automates. Amadio & Meyssonnier [11] ont étudié la décidabilité du problème d'atteignabilité de contrôle pour différents fragments du π-calcul. Ceux-ci démontrent que la combinaison de la génération dynamique de noms avec la mobilité des noms ou le contrôle non borné, mène à un fragment indécidable. D'autre part, ces mêmes auteurs démontrent que la génération dynamique de noms combinée au contexte d'un récipiendaire unique et d'un nombre borné d'entrées, correspond à un fragment décidable du π-calcul.

Hennessy & Riely [59] ont proposé un système de typage pour une extension asynchrone du π-calcul. Une propriété de contrôle d'accès est formalisée à l'aide d'un théorème de sécurité par typage.

Plus récemment, Kremer & Ryan [66] ont utilisé des techniques de vérification basées sur le π-calcul pour analyser un protocole de vote électronique (FOO 92). Ils démontrent que ce protocole satisfait des politiques d'équité, d'éligibilité et de confidentialité qui sont formalisées en terme de propriétés d'atteignabilité et d'équivalence par observation.

2.3.5 Spi calcul

Le *spi calcul* est une extension du π-calcul introduite par Abadi & Gordon [6] et spécialement conçue pour la spécification et l'analyse de protocoles cryptographiques. Sa syntaxe est composée d'un opérateur d'*extraction de couple* (`let` $(x, y) = m$ `in` P), d'un opérateur de *décryptage* (`case` m `of` $\{x\}_k$ `in` P) et d'un opérateur d'*énumération d'entiers* (`case` m `of` $0 : P$ succ$(x) : Q$). Abadi [1] a aussi développé des principes et des règles de typage pour le spi calcul utiles pour la vérification de propriétés de sécurité pour les protocoles cryptographiques. Ces règles « garantissent » qu'un protocole ne divulguera pas ses données secrètes lorsque son typage est correct.

Équivalence de test et bisimulation cadrée. Dans le contexte du spi calcul, Abadi & Gordon [5] ont défini une notion d'*équivalence de test* qui permet de vérifier si deux processus se comportent de façon identique lorsqu'ils sont confrontés aux mêmes environnements hostiles. Cependant, cette formalisation est obscurcie par la présence d'un quantificateur universel qui parcourt l'ensemble des processus observant. Ce problème peut être contourné en considérant la relation de *bisimulation cadrée* (*frame bisimulation*) qui peut être utilisée comme méthode de preuve cohérente, mais malheureusement pas complète, pour la vérification de l'équivalence de test. Abadi & Gordon [5] ont introduit une technique de preuve générale pour l'analyse de protocoles cryptographiques qui est basée sur une spécification dans le spi calcul et sur une validation par l'entremise de l'équivalence de test. En outre, un protocole est démontré sécuritaire s'il est équivalent par test à un autre protocole spécialement construit pour se comporter correctement.

Équivalence de May-testing. À l'aide d'une algèbre de processus similaire au spi calcul, Boreale, De Nicola & Pugliese [21] améliorent la technique de preuve présentée par Abadi & Gordon en substituant l'équivalence de test par une équivalence de *may-testing*. Boreale et al. démontrent que cette dernière est équivalente à l'équivalence de traces entre les processus vus dans leur environnement respec-

tif. L'équivalence de traces peut donc servir de méthode de preuve cohérente et complète pour l'équivalence de *may-testing*, ce qui nous permet de contourner les difficultés présentées par le quantificateur universel qui parcourt l'ensemble des environnements hostiles.

Bisimulation barbelée. En plus de leur équivalence de may-testing, Boreale, De Nicola & Pugliese [21] introduisent une seconde relation d'équivalence sur les processus du spi calcul, appelée *équivalence barbelée* (*barbed equivalence*). Celle-ci est munie d'une méthode de preuve cohérente, mais pas complète, qui consiste à établir une bisimulation entre les processus comparés par rapport à leur environnement respectif (c'est-à-dire leur assignation de valeurs pour leurs variables locales). Cette bisimulation est ensuite vérifiée par des techniques de preuves appelées *techniques up-to*. Boreale et al. utilisent leur méthode de preuve dans le contexte de la validation de protocoles cryptographiques à l'aide de quelques propriétés de sécurité formalisées en terme d'équivalence de may-testing et d'équivalence barbelée. Par exemple, ils définissent la confidentialité comme suit :

> Un protocole *préserve la confidentialité du message* m si, pour tous messages m_1 et m_2, les processus obtenus en substituant toute occurrence de m par, respectivement, m_1 et m_2, sont barbelées équivalents.

2.3.6 Calcul ambiant

Dans le but de modéliser les calculs mobiles, Cardelli & Gordon [29] ont introduit le *calcul ambiant*. Cette algèbre de processus est fondée sur une seule structure qui englobe les agents mobiles, les environnements dans lesquels les agents interagissent et la mobilité de ces environnements. La notion d'environnement (*ambient*) représente les composantes communes aux systèmes distribués telles les messages, les canaux, les noeuds et le code mobile. Les environnements peuvent également être utilisés pour la spécification des dispositifs à calcul mobile, dans lesquels des environnements de calcul sont déplacés. Le calcul ambiant est obtenu d'une extension de la syntaxe du π-calcul et est basé sur deux catégories syntaxiques : les processus

et les capacités. Ces dernières, qui sont composées des capacités d'entrer dans un nom, de sortir d'un nom et d'ouvrir un nom, servent à la modélisation des actions effectuées par les processus. De plus, le calcul ambiant permet la définition d'un processus en tant qu'*environnement* (*ambient*), noté par $n[P]$, où n est un nom et P est un processus évoluant à l'intérieur de l'environnement.

Cardelli, Ghelli & Gordon [28] ont récemment introduit un système de typage pour le calcul ambiant dans le but de modéliser diverses propriétés de sécurité. Cette approche permet la spécification de propriétés de communication et de mobilité par un partitionnement des environnements en ensembles disjoints, appelés *groupes*.

2.4 Méthodes de flots d'information

L'un des principaux enjeux de l'analyse de systèmes informatique consiste à s'assurer qu'aucun programme ne puisse divulguer des informations confidentielles à un tiers partie, que ce soit de façon malicieuse ou par inadvertance. L'analyse des flots d'information aborde ce problème à l'aide de caractérisations précisant lorsque que les flots d'information d'un programme sont sécuritaires, c'est-à-dire lorsque qu'il n'y a aucun flot d'information d'un haut niveau vers un niveau inférieur. La plupart des modèles citées plus haut ne détectent pas les attaques perpétrées par un Cheval de Troie et qui divulguent des données confidentielles à travers un canal clandestin. La détection de ce genre de faille requiert habituellement une analyse minutieuse des flots d'information du protocole ou du système.

Le modèle de Bell-LaPadula [15] fut l'un des premiers modèles développés pour l'analyse de flots d'information dans les systèmes à plusieurs niveaux de sécurité. Ce modèle formalise, entre autres, la manipulation d'objets dans le contexte de ces systèmes à architecture complexe. Sutherland [105,106] a proposé un modèle similaire qui permet une analyse de la sécurité des flots d'information basée sur une propriété appelée *non déductibilité*. Par la suite, McLean [76] a développé une théorie destinée à la formalisation des réseaux de partage de données et basée sur une modélisation de la sécurité en terme de flots de données.

De son côté, Fine [40] a proposé une méthode pour construire des relations d'équivalence propices à l'analyse des flots d'information d'un système. Cette méthode a comme avantage de s'appliquer directement lors des phases de conception et de développement d'un système, plutôt qu'après. Fine [41] propose aussi une technique pour la spécification de propriétés de flots d'information qui combine la *logique temporelle d'actions* (*temporal logic of action*) de Lamport [7] et la *théorie de la composition* (*composition theory*) d'Abadi & Lamport.

2.4.1 Non interférence

Le concept de non interférence [52] nous procure une famille de propriétés de flots d'information qui permettent la détection de canaux clandestins dans les systèmes à architecture à plusieurs niveaux. Intuitivement, la non interférence s'assure qu'un certain groupe d'actions K ne cause pas d'interférence sur un second groupe d'actions L. Cette interférence prend la forme d'une dépendance causale entre une action provenant de L et une action provenant de K. Par exemple, l'interprétation de la non interférence dans le contexte de la confidentialité consiste à vérifier si certaines actions privées causent de l'interférence sur des actions publiques. Cette interprétation nous assure que toute action observée par un participant public est indépendante des actions qui pourraient divulguer de l'information confidentielle. Ainsi, cette propriété de confidentialité nous permet de détecter tout canal clandestin sur lequel le contenu confidentiel d'une action peut être transmis vers un participant de niveau public (e.g. un intrus). Dans le contexte des protocoles d'authentification, la non-interférence permet d'identifier si un intrus est capable de perturber le déroulement du protocole en interagissant avec certains participants.

Non interférence forte non déterministe. Focardi & Gorrieri ont développé un modèle pour l'analyse de protocoles cryptographiques qui est basé sur l'algèbre de processus *Crypographic Security Process Algebra* (CSPA ou CryptoSPA). Cette algèbre de processus permet une spécification formelle des manipulations cryptographiques par l'entremise d'un opérateur de déductions $\vdash_{\text{règle}}$ sur les messages.

Intuitivement, le processus $[\langle a_1 \ldots a_n \rangle \vdash_{\text{règle}} x]P; Q$ désigne un processus qui tente de déduire une information x du tuplet de messages $\langle a_1 \ldots a_n \rangle$ par l'application de la règle $\vdash_{\text{règle}}$. Si le processus réussi sa déduction, alors il évolue selon le processus P ; sinon il évolue selon le processus Q. Focardi & Gorrieri [45] ont introduit la propriété de *non interférence forte non déterministe* (SNNI) qui formalise la confidentialité dans des systèmes à plusieurs niveaux de sécurité. Cette propriété est accompagnée d'une caractérisation algébrique, communément nommée théorème de déroulement (*unwinding theorem*), basée sur une équivalence de traces (entre des processus CCS ou CSPA) et qui peut être utilisée comme algorithme de vérification.

Non déductibilité par composition. Focardi & Martinelli [46–48, 50] ont introduit les propriétés de *non déductibilité par composition* (NDC) et de *non déductibilité par composition généralisée* (GNDC) qui interdisent tout échange d'information allant des participants de haut niveau vers ceux de bas niveau. Cependant, la formalisation de la propriété GNDC contient une quantification universelle qui parcourt l'ensemble des environnements hostiles (modélisés par des processus CSPA), ce qui la rend très difficile à vérifier. Afin de contrer ce problème, Focardi & Martinelli proposent un critère général permettant de vérifier la propriété de GNDC avec un seul processus ennemi (c'est-à-dire un seul environnement hostile) suffisamment puissant. Ce critère peut ainsi servir de condition suffisante dans le cadre d'une caractérisation statique de leur famille de propriétés. Ils donnent une spécification de ce « plus puissant intrus » à l'aide d'un processus CSPA infini qui s'avère être un supremum pour la relation de pré-ordre de traces définie sur tous les processus ennemis CSPA. La propriété de non déductibilité obtenue à l'aide de ce processus ennemi peut être utilisée pour définir des propriétés de sécurité relatives aux protocoles, dont l'authentification de message, la non répudiation des participants à une transaction et l'accord mutuel (*agreement*).

Bugliese, Ceccato & Rossi [27] ont développé une technique de preuve, basée sur le concept de non interférence et l'algèbre de processus CSPA, pour l'analyse de propriétés de *préservation* (*safety*) et de *vivacité* (*liveness*) dans le contexte des

protocoles cryptographiques. Leur approche profite d'une caractérisation sémantique des comportements d'un processus en présence d'un intrus en terme d'un système de transitions étiqueté sensible aux contextes. Leur approche est également basée sur une notion d'équivalence de comportement entre ces systèmes de transitions.

Bien que la non interférence fut principalement formalisée dans des algèbres de processus élémentaires telles CCS, CSP et CSPA, Hennessy & Riely [56,59] offrent une rare formalisation dans le contexte du π-calcul. Celle-ci est accompagnée d'une caractérisation sémantique obtenue à l'aide des relations d'équivalence de *may-testing* et de *must-testing* (définies pour les processus du π-calcul).

2.4.2 Analyse des flots d'information intransitifs

Un désavantage de la non interférence provient du fait qu'exiger l'absence de tout flot d'information déclassifiant est souvent trop drastique pour la spécification de propriétés de sécurité telle la confidentialité. En effet, dans plusieurs cas pratiques, certaines divulgations sont tout simplement inévitables. Afin d'illustrer cette affirmation, considérons un exemple classique qui consiste en un système composé de trois niveaux de sécurité (`privé`, `public` et `déclassification`) qui intègre un protocole cryptographique. Un tel système est présumé confidentiel lorsqu'il satisfait la non interférence entre toute action de niveau `privé`, contenant soit un message secret m ou une clé privée k, et les actions de niveau `public` (que l'on suppose observables par un intrus). Cette vérification de la non interférence doit cependant exclure toute interférence provenant d'une action de niveau `privé` qui fut préalablement déclassifiée par l'entremise d'un système cryptographique. Dans cet exemple, le cryptage de données crée une dépendance causale entre le couple de données secrètes (m, k) et la donnée déclassifiée (message chiffré) $\{m\}_k$. En effet, toute variation dans m ou k est reflétée par $\{m\}_k$, c'est-à-dire toute modification aux valeurs m ou k sera invariablement détectée par un intrus ayant accès au message chiffré $\{m\}_k$. Par contre, cette corrélation est clairement admissible lorsque le mécanisme cryptographique utilisé est suffisamment digne de confiance. Nous

sommes donc à la recherche d'une notion de sécurité qui prend en considération de tels flots d'information admissibles, tout en détectant ceux qui sont inadmissibles.

Afin de répondre à ce besoin, Goguen & Meseguer [52,53] ont introduit la propriété de *non interférence conditionnelle* (CNI). L'idée derrière cette propriété consiste à interdire toute interférence du niveau `privé` sur le niveau `public`, à moins que celle-ci se produise sur un canal contrôlé (e.g. un canal de déclassification). Ce concept de canal de contrôle fut ensuite développé par Rushby [95] qui a introduit le concept de *non interférence intransitive*. Ce concept provient d'un modèle basé sur une relation d'interférence, notée par $\rightsquigarrow$, entre les niveaux de sécurité qui détermine quels niveaux peuvent interférer entre eux. La formalisation des canaux de contrôle dans ce modèle fait en sorte que cette relation d'interférence n'est pas transitive : on peut avoir `privé` $\rightsquigarrow$ `déclassification` et `déclassification` $\rightsquigarrow$ `public`, mais `privé` $\not\rightsquigarrow$ `public`.

Plusieurs formalisations de la non interférence intransitive furent introduites dans la littérature [55,94]. Ces propriétés de flot d'information admettent des caractérisations algébriques locales qui utilisent des notions comme l'observabilité de Boudol [23]. En général, ces caractérisations algébriques stipulent que peu importe l'environnement hostile, aucun changement de comportement du processus (système ou protocole) causé par un certain niveau de sécurité est observable par les niveaux inférieurs, à moins que celui-ci fut préalablement déclassifié (via un niveau de déclassification approprié). Pinsky [90] a proposé un algorithme qui permet de construire une équivalence minimale et à partir de laquelle des conditions de déroulement sont déduites afin de spécifier une politique de déclassification. Nous sommes d'avis que cet algorithme de Pinsky peut être vu comme un algorithme qui construit une certaine bisimulation.

Interférence admissible. Mullins [83] a proposé une formalisation de la non interférence intransitive selon laquelle tout processus P' dérivé du processus P, et n'exécutant aucune action de déclassification, doit satisfaire une même propriété de non interférence. Si nous reformulons cette définition dans le contexte de SNNI

comme propriété de non interférence, nous obtenons la propriété d'*interférence admissible non déterministe* (AI) [83]. Cette propriété admet une caractérisation algébrique (théorème de déroulement) basée sur une équivalence de traces.

Déclassification robuste. Zdancewic & Myers [112] ont introduit un modèle formel pour l'analyse des flots d'information pour les systèmes qui admettent des divulgations d'information intentionnelles. De même, leur modèle comporte une caractérisation de l'information pouvant être divulguée sans compromettre la confidentialité du système. Ces divulgations admissibles de l'information, initiées par des déclassifications, sont ensuite formalisées à partir une propriété de *déclassification robuste* :

> un système est robuste lorsque aucun intrus n'est capable d'exploiter les canaux de déclassification afin d'obtenir des données confidentielles.

Zdancewic & Myers démontrent que tout système satisfaisant des propriétés similaires à la non interférence est nécessairement robuste. Des méthodes de vérification par système de typage pour la propriété de déclassification robuste furent développées par Zdancewic, Myers & Sabelfeld [85,111].

2.5 Méthodes symboliques

Le développement de sémantiques opérationnelles symboliques est souvent un prérequis à la conception d'outils d'analyse automatisée capables de raisonner sur des graphes infinis, sur lesquels les méthodes traditionnelles sont généralement vouées à l'échec. Par exemple, comme nous l'avons constaté plus haut, l'utilisation d'une algèbre de processus avec passage de paramètres lorsque le domaine des messages est infini mène à des graphes de transitions qui ne sont pas à branchement fini, et dans ce cas l'équivalence de bisimulation n'est pas décidable. Une solution élégante à ce problème est proposée par Hennessy & Lin [57], qui définissent une notion de bisimulation symbolique. Cette relation de bisimulation symbolique est basée sur une sémantique symbolique qui permet d'exprimer des processus CCS

avec passage de paramètres en terme de graphes de transitions symboliques finis (c'est-à-dire ayant un nombre fini d'états et de transitions). L'idée fondamentale derrière le modèle de Hennessy-Lin consiste à assigner à chaque action (transition) une formule décrivant les valeurs symboliques (représentées par des variables libres) présentes dans l'action. À partir de ce modèle symbolique, les auteurs introduisent deux généralisations de l'équivalence de bisimulation forte de Milner, appelées *bisimulation à priori* (*early bisimulation*) et *bisimulation à posteriori* (*late bisimulation*).

Rathke [91] a étudié l'utilisation des sémantiques opérationnelles symboliques pour les langages de processus avec passage de paramètres. En utilisant des techniques fondées sur la bisimulation barbelée, il développe de nouvelles notions sémantiques d'équivalence forte et d'équivalence faible. Rathke propose ensuite un modèle cohérent, basé sur un concept de graphe symbolique, permettant la vérification de formules du μ-calcul modal.

À partir d'une algèbre de processus similaire au spi calcul, Boreale [20] introduit une sémantique opérationnelle symbolique fondée sur des techniques d'unification. Boreale propose ensuite une méthode d'analyse de traces qui s'applique directement sur son modèle symbolique. Toujours dans un modèle basé sur le spi calcul, Fiore & Abadi [42] ont proposé un algorithme de décision pour la vérification de connaissances. Leur modèle est également muni d'une procédure symbolique pour l'analyse des connaissances déductibles par un intrus.

2.6 Formalisation du déni de service

Malgré le grand nombre de méthodes proposées dans la littérature pour l'analyse de protocoles de sécurité, la majorité sont essentiellement consacrées à la validation de propriétés de confidentialité et d'authenticité. Jusqu'à tout récemment, très peu d'attention fut accordée à la validation contre les attaques de DoS. Cette incapacité d'établir une définition formelle pour le DoS a fait en sorte que ce type d'attaque est devenu une préoccupation grandissante pour les concepteurs de protocoles.

2.6.1 Modèle de Yu-Gligor

Yu & Gligor [110] ont proposé une spécification formelle et une méthode de vérification pour la prévention des DoS. En utilisant une logique temporelle, ils introduisent des propriétés d'équité, de simultanéité et de « temps-d'attente-fini », dans le contexte d'un modèle général d'allocation de ressources. Yu & Gligor affirment que la formalisation du DoS peut être vu comme un problème de vivacité (*liveness*) :

> certains usagers empêchent d'autres usagers de progresser dans le système pour une durée arbitrairement longue ;

et un problème de préservation (*safety*) :

> certains usagers provoquent la réception de services incorrects à d'autres usagers.

Millen [80] a étendu le modèle d'allocation de ressources de Yu-Gligor en y intégrant une composante de temps. Le modèle de Millen permet une formalisation de propriétés de nature probabilistes pour contrer les DoS, dont une propriété de temps maximal d'attente. Cuppens & Saurel [35] ont proposé une approche similaire qui utilise une propriété d'accessibilité formalisée dans une logique temporelle et une logique déontique. Ces approches développées autour du modèle de Yu-Gligor sont essentiellement fondées sur la propriété de contrôle d'accès appelée *accord de l'usager* (*user agreement*). Cependant, cette propriété de sécurité n'offre aucune protection contre les attaques de DoS perpétrées avant l'authentification mutuelle des participants, comme c'est le cas pour l'attaque *SYN flooding* [101] et les attaques de DoS distribué en général. Dans de telles attaques sur des protocoles qui établissent des canaux de communication authentifiés, l'identité de l'intrus n'est habituellement pas connue puisque la procédure d'authentification n'a pas encore été complétée.

2.6.2 Modèle de Meadows

Meadows [78] a proposé un modèle basé sur les coûts pour l'analyse de la vulnérabilité des protocoles face au DoS. Son approche s'inspire d'une technique de conception de protocoles pour les rendre plus résistants face aux attaques de DoS qui consiste à utiliser une suite de mécanismes d'authentification suivant un ordre croissant de coût ressource (pour chacun des participants) et un ordre croissant de sécurité. Avec cette approche, un intrus doit être disposé à compléter les premières phases du protocole avant qu'il soit en mesure de provoquer d'importantes dépenses de ressources dans des phases avancées du protocole. Plus spécifiquement, la formalisation de Meadows est basée sur le modèle de *protocole échec-arrêt* (*fail-stop protocol*) de Gong-Syverson [54]. Elle formalise ce modèle à l'aide d'un ensemble d'exigences de spécification qui utilisent la relation de pré-ordre causal de Lamport (*causally-precedes relation*). Cette relation détermine comment les événements se précèdent causalement entre eux dans le protocole. L'idée principale derrière ce modèle est fondée sur le principe suivant :

> bien qu'un intrus soit capable de feinter certaines étapes d'authentification (faible) du protocole, pour y arriver il devra dépenser une quantité d'effort qui n'en vaut pas la peine.

Notons que l'outil formel *NRL protocol analyzer* [77] fut utilisé pour illustrer ce modèle.

2.7 Mise en contexte de la thèse

Pour cette thèse, nous avons opté pour un modèle basé sur une algèbre de processus plutôt qu'un modèle de traces. Ce choix fut motivé par l'impossibilité de définir certaines propriétés de flots d'information en termes d'un problème d'atteignabilité. En fait, nous démontrons au Chapitre 5 que les propriétés de non interférence ne sont pas définissables dans le μ-calcul. Une méthode de validation par *équivalence-checking* est donc de mise. D'autre part, nous utilisons une algèbre de processus qui est similaire à CCS plutôt qu'au π-calcul et au spi calcul. La

possibilité de spécifier des processus mobiles est très attrayante, mais la simplicité de CCS, surtout du point de vue sémantique, fut la principale motivation de notre choix. D'autre part, nous introduisons une formalisation du déni de service qui est plus près du modèle de Meadows que de celui de Yu-Gligor. Plus précisément, nous interprétons le modèle de Meadows dans le contexte des algèbres de processus et des propriétés de flots d'information.

CHAPITRE 3

SPÉCIFICATION DE PROTOCOLES DE SÉCURITÉ

Notre premier pas en vue vers la conception d'une méthode de validation de protocoles de sécurité consiste à élaborer un langage dans lequel nous pouvons formaliser à la fois les protocoles et les politiques de sécurité que nous souhaitons faire respecter. Comme nous l'avons constaté dans le Chapitre 2, les algèbres de processus s'avèrent d'excellentes candidates pour cette tâche. Cette approche algébrique à la modélisation des protocoles de sécurité, adoptée par un nombre grandissant d'auteurs et de chercheurs depuis quelques années, consiste à spécifier le protocole comme un regroupement de processus concurrents, chacun représentant un participant au protocole, ayant la capacité de communiquer entre eux afin d'échanger des données. Cependant, la majorité des modèles proposées dans la littérature ne permettent pas une analyse explicite des manipulations cryptographiques.

Dans le spi calcul et l'algèbre de processus CSPA, le cryptage et le décryptage de messages sont effectuées par l'entremise d'un système d'inférence indépendant de la syntaxe, et, par conséquent, ces manipulations ne sont pas directement observables de la sémantique des processus. Par exemple, un participant qui envoie un message m chiffré à l'aide d'une clé k est modélisé par une simple action de sortie « $\overline{c}(\{m\}_k)$ » lorsque le message chiffré $\{m\}_k$ peut être inféré des connaissances courantes du participant. Or, les propriétés de flots d'information, tout particulièrement la non interférence et l'interférence admissible, nécessitent habituellement que ces manipulations soient observables. Suivant cet objectif, nous introduisons l'algèbre de processus *Security Protocols Process Algebra* (SPPA). Dans SPPA, les appels de fonction effectués par les participants sont directement modélisés comme des actions. Ainsi, un participant qui envoie un message m chiffré à l'aide d'une clé k est modélisé par une action « enc_{id} » (où id désigne l'identificateur du participant), suivie par une action de sortie « $\overline{c_{id}}(\{m\}_k)$ ». Ici, l'action « enc_{id} » signifie que le participant associé à l'identificateur id a chiffré un certain message.

Une première version de SPPA fut présentée dans [69]. La syntaxe abstraite de SPPA fut légèrement modifiée de façon à l'uniformiser par rapport à la syntaxe du spi calcul. Ainsi, dans ce chapitre, nous reprenons l'exposé de SPPA tel que présenté dans [71]. D'autre part, l'idée d'introduire un opérateur d'observation directement dans la syntaxe de l'algèbre de processus, de même que le concept de bisimulation par observation, fut initialement proposée dans [84].

3.1 Spécification des messages

Dans cette section, nous présentons un langage de spécification pour les messages. Ce langage est essentiel à notre modèle afin d'établir une représentation formelle des messages échangés lors de protocoles cryptographiques.

3.1.1 Algèbre de messages

Nous considérons l'algèbre de messages suivante qui est fondée sur des catégories syntaxiques disjointes : identificateurs de participant, variables et nombres. Nous désignons par $\mathcal{I}, \mathcal{V}$ et $\mathcal{N}$ les ensembles composés, respectivement, des identificateurs, des variables et des nombres. L'ensemble $\mathcal{T}$ des *termes* de notre algèbre de messages est construit selon la grammaire suivante :

$$\begin{array}{rcl} t & ::= & n \quad (\textit{nombre}) \mid id \quad (\textit{identificateur}) \mid x \quad (\textit{variable}) \\ & \mid & (t,t) \quad (\textit{couple}) \mid \{t\}_t \quad (\textit{cryptage}) \mid [t]_t \quad (\textit{signature}) \\ & \mid & h(t) \quad (\textit{hachage}). \end{array}$$

Étant donné un terme t, nous désignons par $\mathrm{fv}(t)$ l'ensemble des variables présentes dans t. Étant donné une variable $x \in \mathrm{fv}(t)$, $t[a/x]$ désigne le terme obtenu de t en substituant chaque occurrence de la variable x par le terme $a \in \mathit{term}$. Nous utilisons la notation $t[a_1/x_1] \ldots [a_n/x_n]$ afin de désigner une suite de substitutions (lorsqu'une même variable est substituée plus d'une fois, c'est la substitution la plus à gauche qui prévaut).

Définition 3.1. Un terme $t \in \mathcal{T}$ est un *message* s'il est clos, c'est-à-dire $\mathrm{fv}(t) = \emptyset$. L'ensemble de tous les messages est noté par $\mathcal{M}$.

Définition 3.2. Une évaluation est une application $\varrho : \mathcal{V} \rightarrow \mathcal{M}$ qui assigne à chaque variable x un message $\varrho(x)$. Étant donné un terme $t \in \mathcal{T}$, avec $\mathrm{fv}(t) = \{x_1, \ldots, x_n\}$, $\varrho(t)$ désigne le message $t[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$.

Définition 3.3. Nous désignons par $\mathcal{K}$ le sous-ensemble de messages pouvant être utilisés comme clés de cryptage. Pour tout $id \in \mathcal{I}$, l'ensemble $\mathcal{K}_{id}$ désigne le sous-ensemble de $\mathcal{K}$ composé de toute clé initialement connue du participant correspondant à id.

Notons que le contenu de l'ensemble $\mathcal{K}$ dépend largement du système cryptographique utilisé par le protocole à l'étude. Par exemple, dans le cas d'un algorithme de cryptage symétrique par bloc (*symmetric block-cypher*), nous posons $\mathcal{K} = \{k \in \mathcal{N} \mid |k| = N\}$ [1] pour un certain $N \in \mathbb{N}$. De façon plus générale, nous pouvons supposer que $\mathcal{K} = \mathcal{N} \cup \bigcup_{m \geq 1} \{h^m(n) \mid n \in \mathcal{N}\}$.[2] Cependant, afin de simplifier notre présentation, nous supposerons pour cette thèse que $\mathcal{K} = \mathcal{N}$. De plus, de façon à modéliser le cryptage à clé publique, nous utilisons un opérateur idempotent $[-]^{-1} : \mathcal{K} \rightarrow \mathcal{K}$ tel que k^{-1} désigne la clé de décryptage privée correspondant à la clé de cryptage publique k, ou vice versa. Pour le cryptage symétrique, nous posons $k^{-1} = k$. La méthode de spécification présentée dans cette thèse s'inspire du modèle de Dolev-Yao : nous supposons des algorithmes de cryptage et de hachage parfaits. De plus, nous supposons qu'une signature $[a]_k$ est effectuée d'abord par un hachage de a, puis par un cryptage du résultat avec la clé publique k. Ainsi, le terme $[a]_k$ désigne grossièrement le terme $\{h(a)\}_k$.

3.1.2 Fonctions

Nous considérons un ensemble fini $\mathcal{F}$ de *fonctions* $f : \mathcal{M}^n \longrightarrow \mathcal{M}$.[3] Si f est une fonction d'arité $n \geq 1$, nous écrions $f(x_1, \ldots, x_n)$ avec $x_1, \ldots, x_n \in \mathcal{V}$ et nous désignons par $\mathrm{dom}(f)$ son domaine.[4] Par convention, nous écrivons $f(a) = \texttt{fail}$

[1] $|k|$ désigne la longueur (nombre de chiffres) du nombre k.
[2] $h^m(n)$ désigne $h(\ldots h(n) \ldots)$ (m fois).
[3] pour un certain $n \geq 0$, où n est l'arité de f.
[4] $\mathrm{dom}(f) \subseteq \mathcal{M}^n$.

lorsque $a \notin \mathrm{dom}(f)$. Les fonction d'arité $n = 0$ sont appelées *fonction génératrice*. Elles sont essentielles pour la spécification de protocoles de sécurité nécessitant des nonces, des clés fraîches ou des nombres aléatoires. Ce concept de fonction génératrice est également fort utile pour la spécification d'intrus ayant la capacité de créer des messages et des adresses forgés. Les fonctions génératrices sont considérées comme des fonctions pouvant générer des valeurs symboliques sans aucune entrée. Étant donnée une fonction génératrice $new \in \mathcal{F}$, nous écrivons $new(-)$ et nous désignons par $\mathrm{im}(new)$ son image.[5]

À titre d'exemple, voici certaines fonctions que nous utiliserons fréquemment au cours de cette thèse :

- $\mathrm{pair}(x_1, x_2) = (x_1, x_2)$ avec $\mathrm{dom}(\mathrm{pair}) = \mathcal{M} \times \mathcal{M}$;
- $\mathrm{enc}(x_1, x_2) = \{x_2\}_{x_1}$ avec $\mathrm{dom}(\mathrm{enc}) := \mathcal{K} \times \mathcal{M}$;
- $\mathrm{hash}(x_1) = h(x_1)$ avec $\mathrm{dom}(\mathrm{hash}) = \mathcal{M}$;
- $\mathrm{sign}(x_1, x_2) = [x_2]_{x_1}$ avec $\mathrm{dom}(\mathrm{sign}) = \mathcal{K} \times \mathcal{M}$;
- $\mathrm{newMessage}(-)$ avec $\mathrm{im}(\mathrm{newMessage}) = \mathcal{M}$;
- $\mathrm{newNumber}(-)$ avec $\mathrm{im}(\mathrm{newNumber}) = \mathcal{N}$;
- $\mathrm{newId}(-)$ avec $\mathrm{im}(\mathrm{newId}) = \mathcal{I}$;
- $\mathrm{newKey}(-)$ avec $\mathrm{im}(\mathrm{newKey}) = \mathcal{K}$.

Nous étendons la fonction de couplage à tout n-tuplet ; ainsi nous considérons la fonction $\mathrm{pair}(x_1, \ldots, x_n) = (x_1, \ldots, x_n)$.

3.2 Security Protocol Process Algebra

Dans cette section, nous introduisons une algèbre de processus appelée *Security Protocol Process Algebra* (SPPA), qui est une extension de CCS avec passage de paramètres spécialement adaptée à la modélisation de protocoles de sécurité. Notre algèbre de processus permet la spécification d'*appels locaux de fonctions* et introduit la notion d'*actions de marquage*, utilisées pour étiqueter les échanges de messages entre les participants. Notons que l'objectif premier de cette portion de la thèse

[5] $\mathrm{im}(new) \subseteq \mathcal{M}$.

$$
\begin{array}{rcll}
S & ::= & \mathbf{0} & (\textit{nul}) \\
 & | & \overline{c}(t).S & (\textit{sortie}) \\
 & | & c(x).S & (\textit{entrée}) \\
 & | & \texttt{let } x = f(t) \texttt{ in } S & (\textit{appel de fonction}) \\
 & | & \texttt{let } (x,y) = t \texttt{ in } S & (\textit{extraction de couple}) \\
 & | & \texttt{case } t \texttt{ of } \{x\}_{t'} \texttt{ in } S & (\textit{décryptage}) \\
 & | & \texttt{case } t \texttt{ of } [t'']_{t'} \texttt{ in } S & (\textit{vérification de signature}) \\
 & | & [t = t'] \; S & (\textit{comparaison}) \\
 & | & S + S & (\textit{somme}) \\
 & | & S|S & (\textit{produit parallèle})
\end{array}
$$

FIG. 3.1 – Syntaxe abstraite de SPPA.

consiste à établir un modèle générique basé sur une nouvelle algèbre de processus qui convient le plus possible à la méthode de vérification présentée dans cette thèse, et dans lequel il est possible d'interpréter plusieurs modèles déjà proposés dans la littérature.

3.2.1 Syntaxe de SPPA

D'abord, nous considérons un ensemble fini C de *canaux publics*. Les canaux publics sont utilisés pour modéliser les échanges de messages entre les participants. Il y a communément un canal pour chaque étape de l'exécution d'un protocole. Nous pouvons supposer que les canaux publics sont typés, c'est-à-dire à tout canal public c correspond un domaine $\text{dom}(c) \subseteq \mathcal{M}$ spécifiant quels messages peuvent être envoyés ou reçus sur c. Pour les besoins de cette thèse, nous supposons que $\text{dom}(c) = \mathcal{M}$ pour tout $c \in C$.

Les *agents* de SPPA sont construits selon la syntaxe abstraite présentée à la Figure 3.1. Lorsque f est une fonction génératrice, nous écrivons $\texttt{let } x = f(-) \texttt{ in } S$. Nous considérons également, au besoin, une extension de l'opérateur d'extraction de couple $\texttt{let } (x_1, \ldots, x_n) = t \texttt{ in } S$ qui permet le traitement des n-tuplets de messages. Nous utilisons souvent la notation $\sum_{i=1}^{n} S_i$ afin de représenter une somme d'agents $S_1 + \ldots + S_n$.

Afin de prévenir d'éventuels conflits de nom au niveau des variables (par exemple, lorsque nous considérons la somme ou le produit parallèle d'agents ayant les mêmes variables libres), nous ne permettons pas l'utilisation répétée d'une même variable pour définir des agents. Nous devons ainsi renommer les variables au besoin.

Définition 3.4. Une variable x dans l'agent S est *libre* si elle n'est pas dans la portée d'un opérateur d'entrée $c(x)$, d'extraction de couple $\texttt{let}\ (x, y) = t\ \texttt{in}$, d'appel de fonction $\texttt{let}\ x = f(t)\ \texttt{in}$, ou de décryptage $\texttt{case}\ \{t'\}_t\ \texttt{of}\ \{x\}_t\ \texttt{in}$. L'ensemble des variables libres de S est désigné par $\mathrm{fv}(S)$.

Étant donnée une variable libre $x \in \mathrm{fv}(S)$ et un terme t, nous considérons l'opérateur de substitution $S[t/x]$ qui remplace toute occurrence libre de x dans l'agent S par t.

Définition 3.5. Un agent S est *clos* si $\mathrm{fv}(S) = \emptyset$.

Les agents clos sont utilisés pour spécifier les participants des protocoles de sécurité.

Définition 3.6. Un *participant* SPPA est un couple (S, id) composé d'un agent clos S et d'un identificateur $id \in \mathcal{I}$.

La motivation derrière cette notation est d'établir un lien entre, d'une part un agent S et ses sous-agents, et d'autre part leur unique propriétaire (participant), via son identificateur id. Par abus de notation, nous utilisons souvent l'expression A à la fois comme référence à l'entité (participant) qui participe au protocole et comme référence au participant SPPA correspondant. Dans ce cas, nous posons $A = (S_A, id_A)$, où S_A désigne l'*agent initial* de A, c'est-à-dire l'agent clos spécifiant le comportement intégral du participant A dans le protocole. De plus, nous utilisons fréquemment l'identificateur id_A en tant que message contenant l'adresse de A. Afin d'alléger la présentation, nous adoptons la notation suivante : étant donnés les participants SPPA $A_1 = (S_1, id)$ et $A_2 = (S_2, id)$ (ils doivent avoir le même identificateur), nous écrivons

$\mathbf{0}$	à la place de	$(\mathbf{0}, id)$;
$\overline{c}(t).A_1$	à la place de	$(\overline{c}(t).S_1, id)$;
$c(x).A_1$	à la place de	$(c(x).S_1, id)$;
$\texttt{let}\ x = f(t)\ \texttt{in}\ A_1$	à la place de	$(\texttt{let}\ x = f(t)\ \texttt{in}\ S_1, id)$;
$\texttt{let}\ (x, y) = t\ \texttt{in}\ A_1$	à la place de	$(\texttt{let}\ (x, y) = t\ \texttt{in}\ S_1, id)$;
$\texttt{case}\ t\ \texttt{of}\ \{x\}_{t'}\ \texttt{in}\ A_1$	à la place de	$(\texttt{case}\ t\ \texttt{of}\ \{x\}_{t'}\ \texttt{in}\ S_1, id)$;
$\texttt{case}\ t\ \texttt{of}\ [t'']_{t'}\ \texttt{in}\ A_1$	à la place de	$(\texttt{case}\ t\ \texttt{of}\ [t'']_{t'}\ \texttt{in}\ S_1, id)$;
$[t = t']A_1$	à la place de	$([t = t']S_1, id)$;
$A_1 + A_2$	à la place de	$(S_1 + S_2, id)$;
$A_1 \vert A_2$	à la place de	$(S_1 \vert S_2, id)$;

Cette nouvelle notation offre une correspondance directe entre un participant SPPA $A = (S, id)$ et son agent S. De plus, afin d'avoir facilement accès aux identificateurs, nous désignons l'identificateur à l'intérieur du participant SPPA A par id_A.

Dans le but de modéliser des protocoles de sécurité, nous utilisons une approche classique qui consiste à spécifier les participants d'un protocole en terme de processus concurrents. Les *processus* SPPA sont construits comme suit :

$$\begin{aligned} P \ ::= \quad & A \quad (\textit{participant}) \ \mid \ A \parallel P \quad (\textit{protocole}) \\ \mid \ & P\backslash L \quad (\textit{restriction}) \ \mid \ P/\mathcal{O} \quad (\textit{observation}). \end{aligned}$$

où A est un participant SPPA, L est un ensemble et $\mathcal{O}$ est une application (ces deux derniers éléments syntaxiques sont clarifiés aux Sections 3.2.3 et 3.2.4) Notons que nous écrivons souvent $P \setminus (L \cup L')$ à la place de $(P \setminus L) \setminus L'$. La communication entre les participants est formalisée par l'opérateur $\parallel$.

Remarque 3.7. Afin de s'assurer que les processus SPPA conservent leur nature finie, nous devons établir les restrictions syntaxiques suivantes (celles-ci sont souvent appliquées sur d'autres algèbres de processus pour des raisons similaires). Nous ne permettons pas les définitions récursives de la forme $P ::= P_1 \setminus L$, $P ::= P_1/\mathcal{O}$, $P ::= P_1 \vert P_2$ ou $P ::= P_1 \parallel P_2$ telles que P apparaît quelque part dans la définition de P_1 ou de P_2. Ainsi, nous supposons que toute définition récursive d'un agent ou d'un processus SPPA n'utilise jamais un opérateur de restriction, ni un opérateur

d'observation, ni un opérateur de composition parallèle, ni un opérateur de protocole. De telles définitions récursives mènent souvent à des processus infinis. Par exemple, les processus $P ::= (c(x).P) \setminus L$ et $P ::= P|P'$ sont exclus, tandis que les processus $P ::= \texttt{let}\ x = f(t)\ \texttt{in}\ P$ et $P ::= c(x).P + P'$ sont conservés.

Exemple 3.8 (Spécification du protocole de Needham-Schröeder). Dans cet exemple, nous offrons une spécification formelle, à l'aide de SPPA, du protocole d'authentification à clé publique de Needham-Schröeder, dont la spécification Alice et Bob est présentée à la Section 1.2.1. Ce protocole possède deux participants, A et B, qui sont respectivement spécifiés par les participants SPPA $A = (S_A, id_A)$ et $B = (S_B, id_B)$, où id_A et id_B sont les identificateurs respectifs des participants, et les agents initiaux S_A et S_B sont définis comme suit :

$$
\begin{aligned}
S_A ::= \sum_{id \in \mathcal{I}} & \texttt{let}\ x_1 = \text{newNumber}(-)\ \texttt{in}\ \texttt{let}\ x_2 = \text{pair}(x_1, id_A)\ \texttt{in} \\
& \texttt{let}\ x_3 = \text{enc}(k_{id}, x_2)\ \texttt{in}\ \texttt{let}\ x_4 = \text{pair}(id_A, id, x_3)\ \texttt{in} \\
& \overline{c_{1id}}(x_4).\ c_{2id_A}(x_5).\ \texttt{let}\ (x_6, x_7, x_8) = x_5\ \texttt{in} \\
& [x_6 = id]\ [x_7 = id_A]\ \texttt{case}\ x_8\ \texttt{of}\ \{x_9\}_{k_A}\ \texttt{in} \\
& \texttt{let}\ (x_{10}, x_{11}) = x_9\ \texttt{in}\ [x_{10} = x_1]\ \texttt{let}\ x_{12} = \text{auth}(id)\ \texttt{in} \\
& \texttt{let}\ x_{13} = \text{enc}(k_{id}, x_{11})\ \texttt{in}\ \texttt{let}\ x_{14} = \text{pair}(id_A, id, x_{13})\ \texttt{in} \\
& \overline{c_{3id}}(x_{14}).\mathbf{0}
\end{aligned}
$$

$$
\begin{aligned}
S_B ::= \ & c_{1id_B}(y_1).\ \texttt{let}\ (y_2, y_3, y_4) = y_1\ \texttt{in}\ [y_3 = id_B]\ \texttt{case}\ y_4\ \texttt{of}\ \{y_5\}_{k_B}\ \texttt{in} \\
& \texttt{let}\ (y_6, y_7) = y_5\ \texttt{in}\ [y_7 = y_2]\ \texttt{let}\ y_8 = \text{newNumber}(-)\ \texttt{in} \\
& \texttt{let}\ y_9 = \text{pair}(y_6, y_8)\ \texttt{in}\ \texttt{let}\ y_{10} = \text{enc}(k_{id_A}, y_9)\ \texttt{in} \\
& \texttt{let}\ y_{11} = \text{pair}(id_B, y_2, y_{10})\ \texttt{in} \\
& \overline{c_{2y_2}}(y_{11}).\ \texttt{let}\ (y_{13}, y_{14}, y_{15}) = y_{12}\ \texttt{in} \\
& c_{3B}(y_{12}).\ [y_{14} = id_B]\ [y_{13} = y_2]\ \texttt{case}\ y_{15}\ \texttt{of}\ \{y_{16}\}_{k_B}\ \texttt{in} \\
& [y_{16} = y_8]\ \texttt{let}\ y_{17} = \text{auth}(y_2)\ \texttt{in}\ \mathbf{0}
\end{aligned}
$$

où c_{jid} désigne le canal public utilisé pour envoyer le $j^{\text{ième}}$ message du protocole au participant dont l'identificateur est id. L'ensemble des canaux publics est donné

par $C = \bigcup_{id \in \mathcal{I}} \{c_{1id}, c_{2id}, c_{3id}\}$. Notons qu'en vue d'obtenir une spécification finie de l'agent S_A, nous devons supposer que l'ensemble $\mathcal{I}$ est fini, ce qui nous assure que la somme $\sum_{id \in \mathcal{I}}$ correspond bien à une somme finie. D'autre part, cette spécification utilise une fonction d'authentification auth définie par $\text{auth}(id) = id$ si $id \in \mathcal{I}$, et telle que $\text{dom}(\text{auth}) = \mathcal{I}$. L'ensemble des fonctions est donné par $\mathcal{F} = \{\text{newNumber, pair, enc, auth}\}$.

Le protocole de Needham-Schröeder est donc spécifié par le processus SPPA $P ::= A \parallel B$ dans lequel les participants A et B interagissent.

3.2.2 Actions

Les *actions* de SPPA sont définies comme suit :

$$
\begin{array}{rcll}
\alpha & ::= & \overline{c_{id}}(a) & (\textit{sortie}) \\
& | & c_{id}(a) & (\textit{entrée}) \\
& | & \text{split}_{id} & (\textit{extraction}) \\
& | & f_{id} & (\textit{appel de fonction}) \\
& | & \text{dec}_{id} & (\textit{décryptage}) \\
& | & \text{signv}_{id} & (\textit{vérification de signature}) \\
& | & \texttt{fail}^{\texttt{f}}_{\texttt{id}} & (\textit{échec de fonction}) \\
& | & \texttt{fail}^{\texttt{dec}}_{\texttt{id}} & (\textit{échec de décryptage}) \\
& | & \texttt{fail}^{\texttt{signv}}_{\texttt{id}} & (\textit{échec de vérification de signature}) \\
& | & \texttt{fail}^{=}_{\texttt{id}} & (\textit{échec d'une comparaison}) \\
& | & \delta(a) & (\textit{marquage}) \\
& | & \tau & (\textit{interne})
\end{array}
$$

où $a \in \mathcal{M}$ est un message. Par exemple, l'action d'appel de fonction enc_{id_A} signifie que le participant A a chiffré un certain message a avec une clé k (quelconque) ; l'action de sortie $\overline{c_{id_A}}(\{a\}_k)$ signifie que le participant A a envoyé le message $\{a\}_k$ sur canal public c ; et l'action de décryptage dec_{id_A} signifie que le participant A a décrypté, avec succès, un certain message $\{a\}_k$. Notons que l'action interne τ est principalement utilisée pour exprimer des comportements non observables.

Dans la plupart des algèbres de processus avec passage de paramètres, la communication est exprimée en remplaçant des actions de sortie et d'entrée conjuguées ($\overline{c}(a)$ et $c(a)$) par l'action interne τ. Cependant, cette approche à la modélisation de la communication cause une importante perte d'information au niveau du contenu des valeurs échangées et des participants impliqués. Les actions de marquage sont introduites dans le but d'établir une annotation sur la sémantique des processus SPPA ; ils n'apparaissent pas au niveau de la syntaxe des processus puisque qu'ils ne sont pas considérées comme étant des préfixes, et leur sémantique particulière restreint leur occurrence de façon à étiqueter les communications entre les participants. Une action de marquage est composé de trois paramètres : un identificateur, un canal et un message. Intuitivement, l'occurrence d'un *marqueur de sortie* $\overline{\delta^c_{id_A}}(a)$ signifie que « *le participant A a envoyé le message a sur le canal c* », et l'occurrence d'un *marqueur d'entrée* $\delta^c_{id_A}(a)$ signifie que « *le participant A a reçu le message a sur le canal c* ».

Définition 3.9. L'ensemble de toutes les actions est désignée par Act. Pour tout participant A, nous considérons l'ensemble Act_A des actions exécutables seulement par A, défini par :

$$\begin{aligned} Act_A \;=\; & \{\overline{c_{id_A}}(a),\ c_{id_A}(a),\ \overline{\delta^c_{id_A}}(a),\ \delta^c_{id_A}(a) \in Act \mid c \in C \text{ et } a \in \mathcal{M}\} \\ & \cup \{f_{id_A},\ \mathtt{fail}^{\mathtt{f}}_{\mathtt{id_A}} \mid \mathtt{f} \in \mathcal{F}\} \\ & \cup \{\mathrm{dec}_{id_A},\ \mathrm{signv}_{id_A},\ \mathtt{fail}^{\mathrm{dec}}_{\mathtt{id_A}},\ \mathtt{fail}^{\mathrm{signv}}_{\mathtt{id_A}},\ \mathtt{fail}^{=}_{\mathtt{id_A}}\} \end{aligned}$$

Notons que nous utilisons C pour désigner à la fois l'ensemble des canaux publics et l'ensemble des actions de sortie et d'entrée. Nous considérons aussi l'ensemble des actions visibles défini par $Vis = Act \setminus \{\tau\}$.

3.2.3 Critère d'observation

Boudol [23] a introduit une notion de critère d'observation permettant une formalisation de l'observation d'une suite d'actions dans le but d'établir une relation d'équivalence entre les processus qui reflète le point de vue d'un observateur. Un

tel critère d'observation sur l'ensemble Act des actions est donc déterminé par la donnée d'un sous-ensemble $L \subseteq Act$ d'actions observables.

Définition 3.10. Un *critère d'observation* est une application

$$\mathcal{O} : Act^* \longrightarrow \wp(Act^*)\ ^6$$

qui satisfait les conditions suivantes :

1. $\tau^n \in \mathcal{O}(\epsilon)\ ^7$ pour tout $n \geq 0$, et
2. $\mathcal{O}(\gamma\alpha) = \begin{cases} \mathcal{O}(\gamma) & \text{si } \epsilon \in \mathcal{O}(\alpha) \\ \{\gamma'\alpha\tau^n \mid \gamma' \in \mathcal{O}(\gamma) \text{ et } n \geq 0\} & \text{sinon.} \end{cases}$

Si $\epsilon \notin \mathcal{O}(\alpha)$, alors nous disons que l'action α est observable selon le critère d'observation $\mathcal{O}$. Dans ce cas, $\mathcal{O}(\alpha)$ est composé de toutes les suites d'actions correspondant à la même observation $\alpha \in Vis$. En particulier, nous devons avoir $\alpha \in \mathcal{O}(\alpha)$. D'autre part, si $\epsilon \in \mathcal{O}(\alpha)$, alors l'action α n'est pas observable selon $\mathcal{O}$. Dans ce cas, nous avons $\mathcal{O}(\alpha) = \mathcal{O}(\epsilon)$. Entre autres, nous avons toujours $\mathcal{O}(\tau) = \mathcal{O}(\epsilon)$. Dans la proposition suivante, nous énonçons quelques propriétés fondamentales des critères d'observation. Les preuves de ces propriétés se déduisent facilement de la Définition 3.10.

Proposition 3.11. *Soit $\mathcal{O}$ un critère d'observation et considérons la suite d'actions $\gamma = \alpha_1 \ldots \alpha_n \in Act^*$ telle que $\epsilon \notin \mathcal{O}(\alpha_{k_j})$ pour $j = 1, \ldots, m$ et $\epsilon \in \mathcal{O}(\alpha_i)$ pour tout $i \notin \{k_1, \ldots, k_m\}$.*

1. *Si $\gamma' \in \mathcal{O}(\gamma)$, alors $\gamma' = \tau^{l_0}\alpha_{k_1}\tau^{l_1} \ldots \tau^{l_{m-1}}\alpha_{k_m}\tau^{l_m}$ pour certains entiers $l_0, l_1, \ldots, l_m \geq 0$.*
2. *Nous avons $\mathcal{O}(\gamma) = \mathcal{O}(\alpha_{k_1} \ldots \alpha_{k_m})$.*

L'observation d'une suite d'actions correspond donc à sa sous-suite d'actions observables. Ainsi, étant donné un critère d'observation, la relation $\mathcal{O}(\gamma) = \mathcal{O}(\gamma')$

[6] $\wp(E)$ désigne l'ensemble des parties de l'ensemble E.
[7] ϵ désigne la suite vide.

entre les suites d'actions nous procure une relation d'équivalence sur Act^* : deux suites d'actions sont équivalentes dès qu'elles correspondent à la même suite minimale d'actions visibles $\alpha_{k_1} \dots \alpha_{k_m}$.

Définition 3.12. Soit $L \subseteq Vis$ un sous-ensemble d'actions visibles. Le critère d'observation $\mathcal{O}_L$ est défini comme suit :

1. $\mathcal{O}_L(\epsilon) = (Act \setminus L)^*$, et
2. $\mathcal{O}_L(\gamma\alpha) = \begin{cases} \mathcal{O}_L(\gamma) & \text{si } \alpha \in Act \setminus L \\ \{\gamma'\alpha\gamma'' \mid \gamma' \in \mathcal{O}_L(\gamma) \text{ et } \gamma'' \in (Act \setminus L)^*\} & \text{si } \alpha \in L \end{cases}$

Seuls les comportements provenant de l'ensemble L sont observables par rapport au critère d'observation $\mathcal{O}_L$. En particulier, deux suites d'actions procurent la même observation par rapport au critère faible $\mathcal{O}_{Vis}$ si leur contenu visible est identique. Dans le contexte de l'analyse des protocoles de sécurité, nous sommes tout particulièrement intéressés à la famille de critères d'observation $\{\mathcal{O}_{id}\}_{id \in \mathcal{I}}$, où $\mathcal{O}_{id} = \mathcal{O}_{Act_A \cup C}$ lorsque $id = id_A$. Ainsi, le critère d'observation $\mathcal{O}_{id_A}$, noté simplement par $\mathcal{O}_A$, décrit les actions observables par le participant A.

Proposition 3.13. *Soit $L \subseteq Vis$ et soit $\gamma \in Act^*$. Pour toutes actions $\alpha_1, \alpha_2 \in L$ telles que $\alpha_1 \neq \alpha_2$, si $\alpha_1 \in \mathcal{O}_L(\gamma)$, alors $\alpha_2 \notin \mathcal{O}_L(\gamma)$.*

Démonstration. Cet énoncé est une conséquence directe du fait que la suite d'actions $\alpha_{k_1} \dots \alpha_{k_m} \in \mathcal{O}_L(\gamma)$, donnée par la Proposition 3.11, est unique. □

3.2.4 Sémantique de SPPA

La sémantique opérationnelle d'un processus SPPA est présentée aux Figures 3.2 et 3.3, où $a \in \mathcal{M}$ est un message, $L \subseteq Act$ est un sous-ensemble d'actions, A, A', B, B' sont des participants SPPA et P, P', Q, Q' sont des processus SPPA.

La règle `Sortie` permet l'envoie de messages sur un canal public, tandis que la règle `Entrée` doit considérer tous les messages possibles pouvant être reçus sur un canal public. La règle `Extraction` permet l'extraction de couples, et s'étend de façon naturelle à l'extraction de n-tuplets. La règle `Fonction` permet l'exécution

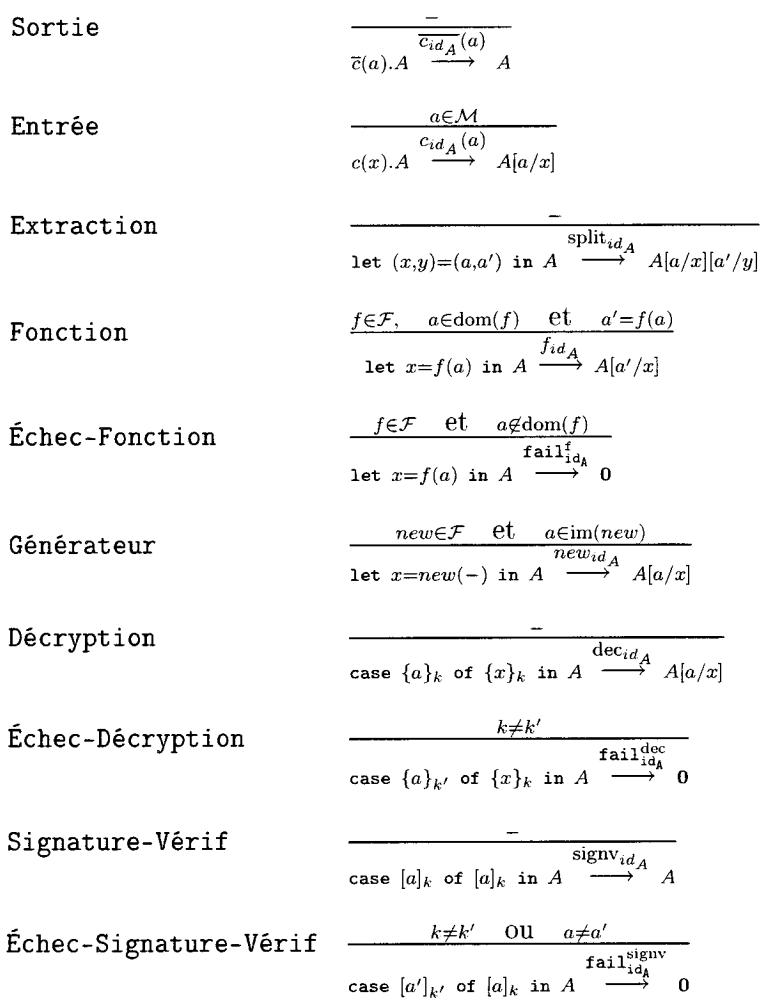

FIG. 3.2 – Sémantique opérationnelle des processus SPPA.

Comparaison $\dfrac{A \xrightarrow{\alpha} A'}{[a{=}a]\ A \xrightarrow{\alpha} A'}$

Échec-Comparaison $\dfrac{a \neq a'}{[a{=}a']\ A \xrightarrow{\mathtt{fail}^{=}_{\mathrm{id}_A}} \mathbf{0}}$

Somme $\dfrac{A \xrightarrow{\alpha} A'}{A{+}B \xrightarrow{\alpha} A'}$ et $\dfrac{B \xrightarrow{\alpha} B'}{A{+}B \xrightarrow{\alpha} B'}$

Parallèle $\dfrac{A \xrightarrow{\alpha} A'}{A|B \xrightarrow{\alpha} A'|B}$ et $\dfrac{B \xrightarrow{\alpha} B'}{A|B \xrightarrow{\alpha} A|B'}$

Protocole $\dfrac{P \xrightarrow{\alpha} P' \quad \text{et} \quad \alpha \notin C}{P\|Q \xrightarrow{\alpha} P'\|Q}$ et $\dfrac{Q \xrightarrow{\alpha} Q' \quad \text{et} \quad \alpha \notin C}{P\|Q \xrightarrow{\alpha} P\|Q'}$

Synchronisation $\dfrac{P \xrightarrow{\overline{c_{id}}(a)} P' \quad \text{et} \quad Q \xrightarrow{c_{id'}(a)} Q'}{P\|Q \xrightarrow{\overline{\delta^c_{id}}(a)} P'\#Q \xrightarrow{\delta^c_{id'}(a)} P'\|Q'}$ et

$\dfrac{Q \xrightarrow{\overline{c_{id'}}(a)} Q' \quad \text{et} \quad P \xrightarrow{c_{id}(a)} P'}{P\|Q \xrightarrow{\overline{\delta^c_{id'}}(a)} P\#Q' \xrightarrow{\delta^c_{id}(a)} P'\|Q'}$

Restriction $\dfrac{P \xrightarrow{\alpha} P' \quad \text{et} \quad \alpha \notin L}{P\backslash L \xrightarrow{\alpha} P'\backslash L}$

Observation $\dfrac{P \xrightarrow{\gamma} P' \quad \text{et} \quad \alpha \in \mathcal{O}(\gamma)}{P/\mathcal{O} \xrightarrow{\alpha} P'/\mathcal{O}}$

FIG. 3.3 – Sémantique opérationnelle des processus SPPA.

d'un appel de fonction par un participant. La règle `Décryption` permet le décryptage des messages chiffrés et la règle `Signature-Vérif` permet la validation de messages signés. Notons que la validation d'un message signé ne permet pas pour autant de récupérer le message à l'intérieur car nous avons supposé qu'il est haché. Les règles `Échec-Fonction`, `Échec-Décryption` et `Échec-Signature-Vérif` traitent les cas où, respectivement, une fonction est appelée sur un message hors de son domaine, un décryptage est tenté avec une mauvaise clé, et une vérification de signature échoue (soit à cause d'une mauvaise clé, ou d'un message incorrect). La règle `Comparaison` permet la vérification d'une égalité entre deux messages, tandis que la règle `Échec-Comparaison` traite les cas où l'égalité n'est pas satisfaite. Notons que toute action d'échec mène au processus nul $\mathbf{0}$, bien que nous pourrions définir des processus spéciaux pour les différents types d'échec.

Les règles `Somme` et `Parallèle` permettent, respectivement, la formalisation de la somme non déterministe et le produit parallèle de participants SPPA (ayant le même identificateur). Les règles `Protocole` et `Synchronisation` permettent la formalisation de protocoles, où l'opérateur $\|$ est similaire au produit parallèle entre participants et dans lequel la communication entre les participants est accomplie (et forcée) à l'aide des canaux publics. Lorsque deux processus P et Q communiquent, ils entrent dans un état intermédiaire $P\#Q$ (un état « occupé ») qui ne peut pas être atteint autrement. Nous pouvons donc considérer $P\#Q$ comme étant un processus SPPA ayant une unique transition entrante $P \parallel Q \xrightarrow{\overline{\delta^c_{id}}(a)} P\#Q$, et une unique transition sortante $P\#Q \xrightarrow{\delta^c_{id'}(a)} P' \parallel Q'$. Cette formalisation de la communication nous assure, entre autres, que toute action de marquage de sortie est nécessairement suivie d'une action de marquage d'entrée. Ainsi, l'état $P\#Q$ n'est pas accessible autrement.

La règle `Restriction` interprète $P \setminus L$ (où L est un sous-ensemble d'actions) comme P avec les actions dans L retirées. Finalement, la règle `Observation` interprète l'observation d'un processus par rapport à un critère d'observation $\mathcal{O}$, où le *calcul* $P \xrightarrow{\gamma} P'$, pour une suite d'actions $\gamma = \alpha_0\alpha_1 \ldots \alpha_n \in Act^*$, désigne la suite finie de transitions $P \xrightarrow{\alpha_0} P_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} P'$. Ainsi, étant donné $L \subseteq Vis$, $P/\mathcal{O}_L$ cor-

respond à P avec les actions à l'extérieur de L ignorées (c'est-à-dire remplacées par l'action τ). Le concept de $\mathcal{O}$-observation d'un processus vise à définir le processus obtenu de son observation par rapport au critère d'observation $\mathcal{O}$.

Définition 3.14. Le processus P' est une *dérivée* du processus P s'il existe un calcul $P \xrightarrow{\gamma} P'$ pour un certain $\gamma \in Act^*$. L'ensemble des dérivées de P est donné par

$$\mathcal{D}(P) = \{P' \mid \exists_{\gamma \in Act^*}\ P \xrightarrow{\gamma} P'\}.$$

Pour toute suite d'actions visibles $\gamma = \alpha_1 \ldots \alpha_n \in Vis^*$, nous écrivons $P \stackrel{\gamma}{\Longrightarrow} P'$ dès que $P \xrightarrow{\gamma'} P'$ avec $\gamma' = \tau^{k_0}\alpha_1\tau^{k_1} \ldots \tau^{k_{n-1}}\alpha_n\tau^{k_n}$ pour certains $k_0, \ldots, k_n \geq 0$.

Remarque 3.15. L'aspect important apporté par les restriction syntaxiques imposées à la Remarque 3.7 est de s'assurer que tout processus SPPA ne possède qu'un nombre fini de dérivées, à moins que celles-ci ne soient obtenues d'une application des règles `Entrée` ou `Générateur` de la sémantique de SPPA.

Exemple 3.16. Considérons les participants SPPA $A = (S_A, id_A)$ et $B = (S_B, id_B)$ où les agents initiaux S_A et S_B sont définis par :

$$S_A ::= \overline{c}(a).\mathbf{0} \quad \text{et} \quad S_B ::= c(x).\mathbf{0}$$

avec $a \in \mathcal{M}$. La sémantique du processus $P ::= A \parallel B$ est illustrée à la Figure 3.4.

$$A \parallel B \xrightarrow{\overline{\delta^c_{id_A}}(a)} A\#B \xrightarrow{\delta^c_{id_B}(a)} \mathbf{0} \parallel \mathbf{0}$$

FIG. 3.4 – Sémantique du processus $A \parallel B$.

Exemple 3.17. Considérons les sous-ensembles d'actions $K, L \subseteq Vis$ tels que $L = \{\beta_1, \beta_2, \beta_3\}$ et $K = L \cup \{\alpha\}$, ainsi que les critères d'observation $\mathcal{O}_K$ et $\mathcal{O}_L$. À la Figure 3.5, nous illustrons la sémantique d'un processus SPPA P et celles de

processus observés $P/\mathcal{O}_K$ et $P/\mathcal{O}_L$. Notons que, dans la sémantique de ces deux derniers processus, nous avons omis, pour chaque état, une transition étiquetée par τ qui boucle.

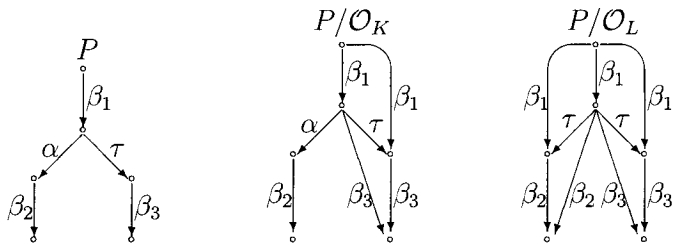

FIG. 3.5 – Processus observés.

3.3 Relations d'équivalence sur les processus

La bisimulation offre une sémantique expressive pour les processus à partir d'une équivalence par rapport à leurs comportements.

Définition 3.18 (Bisimulation). Soient P et Q des processus. Une *bisimulation* entre P et Q est une relation $\mathcal{R} \subseteq \mathcal{D}(P) \times \mathcal{D}(Q)$ telle que

- $(P, Q) \in \mathcal{R}$;
- Pour toute action $\alpha \in Act$, si $(P_1, Q_1) \in \mathcal{R}$ et $P_1 \xrightarrow{\alpha} P_2$, alors il existe une transition $Q_1 \xrightarrow{\alpha} Q_2$ telle que $(P_2, Q_2) \in \mathcal{R}$;
- Pour toute action $\alpha \in Act$, si $(P_1, Q_1) \in R$ et $Q_1 \xrightarrow{\alpha} Q_2$, alors il existe une transition $P_1 \xrightarrow{\alpha} P_2$ telle que $(P_2, Q_2) \in \mathcal{R}$.

Si une telle bisimulation entre P et Q existe, alors on écrit $P \simeq Q$.

Un algorithme de complexité temporelle polynomiale (PTIME) pour vérifier la bisimulation entre deux processus dotés d'une sémantique finie (c'est-à-dire un graphe de transitions fini) est donné par Milner [81].

Si nous sommes seulement intéressés à comparer les comportements visibles des processus, donc en omettant les occurrences de l'action τ, nous obtenons une extension de l'équivalence de bisimulation appelée *bisimulation faible* et notée par $\simeq_{\text{faible}}$.

La définition de cette relation s'obtient de celle de la bisimulation en remplaçant les transitions $\xrightarrow{\alpha}$ par $\stackrel{\alpha}{\Longrightarrow}$. Afin de distinguer ces deux notions, nous appelons souvent la bisimulation par bisimulation forte.

3.4 Bisimulation par observation

Le concept de $\mathcal{O}$-bisimulation[8] formalise la notion d'indistinguabilité de comportement par rapport à un critère d'observation $\mathcal{O}$.

Définition 3.19. Soient P et Q des processus, et soit $\mathcal{O}$ un critère d'observation. Une *$\mathcal{O}$-bisimulation* entre P et Q est une relation $R \subseteq \mathcal{D}(P) \times \mathcal{D}(Q)$ telle que

- $(P, Q) \in R$,
- Pour toute action $\alpha \in Act$, si $(P_1, Q_1) \in R$ et $P_1 \xrightarrow{\alpha} P_2$, alors il existe une transition $Q_1 \xrightarrow{\gamma} Q_2$, avec $\alpha \in \mathcal{O}(\gamma)$, telle que $(P_2, Q_2) \in \mathcal{R}$.
- Pour toute action $\alpha \in Act$, si $(P_1, Q_1) \in R$ et $Q_1 \xrightarrow{\alpha} Q_2$, alors il existe une transition $P_1 \xrightarrow{\gamma} P_2$, avec $\alpha \in \mathcal{O}(\gamma)$, telle que $(P_2, Q_2) \in \mathcal{R}$.

Dans ce cas, nous écrivons $P \simeq_{\mathcal{O}} Q$ et nous disons que P et Q sont *$\mathcal{O}$-bisimilaires.*

Nous pouvons facilement constater que la $\mathcal{O}_{Vis}$-bisimulation correspond à la bisimulation faible. D'autre part, si nous remplaçons dans la Définition 3.19 le critère d'observation $\mathcal{O}$ par l'application identité $\mathcal{O}_{Act}$[9], alors nous obtenons le critère fort à travers lequel toute suite d'actions est observable et, conséquemment, distinguable l'une de l'autre. La $\mathcal{O}_{Act}$-bisimulation correspond donc à la bisimulation forte. De façon plus générale, nous avons la caractérisation suivante de la $\mathcal{O}$-bisimulation en terme de bisimulation.

Proposition 3.20. *Soient P et Q des processus et soit $\mathcal{O}$ un critère d'observation. Alors, $P \simeq_{\mathcal{O}} Q$ si et seulement si $P/\mathcal{O} \simeq Q/\mathcal{O}$.*

Démonstration. D'abord, supposons que $P \simeq_{\mathcal{O}} Q$ et soit $R \subseteq \mathcal{D}(P) \times \mathcal{D}(Q)$ une relation qui témoigne de ce fait, c'est-à-dire qui satisfait les conditions exposées à la

[8] appelé $\mathcal{O}$-congruence par Boudol [23].
[9] définie par $\mathcal{O}_{Act}(\alpha) = \alpha$ pour toute action $\alpha \in Act$.

Définition 3.19. Nous démontrons que la relation $R' = \{(P'/\mathcal{O}, Q'/\mathcal{O}) \mid (P', Q') \in R\}$ est une bisimulation entre $P/\mathcal{O}$ et $Q/\mathcal{O}$, donc $P/\mathcal{O} \simeq Q/\mathcal{O}$. Nous commençons par constater que $(P/\mathcal{O}, Q/\mathcal{O}) \in R'$. Ensuite, considérons $(P_1/\mathcal{O}, Q_1/\mathcal{O}) \in R'$ (d'où $(P_1, Q_1) \in R$) avec $P_1/\mathcal{O} \xrightarrow{\alpha} P_2/\mathcal{O}$ (la preuve pour le cas où $Q_1/\mathcal{O} \xrightarrow{\alpha} Q_2/\mathcal{O}$ est analogue). Or, par la règle `Observation`, il existe une suite d'actions γ telle que $\alpha \in \mathcal{O}(\gamma)$ et $P_1 \xrightarrow{\gamma} P_2$. Mais, puisque $(P_1, Q_1) \in R$, nous avons $Q_1 \xrightarrow{\gamma} Q_2$ avec $(P_2, Q_2) \in R$, pour un certain Q_2. Ainsi, par la règle `Observation`, nous voyons que $Q_1/\mathcal{O} \xrightarrow{\alpha} Q_2/\mathcal{O}$, avec $(P_2/\mathcal{O}, Q_2/\mathcal{O}) \in R'$ car $(P_2, Q_2) \in R$.

Réciproquement, supposons que $P/\mathcal{O} \simeq Q/\mathcal{O}$ et soit $R' \subseteq \mathcal{D}(P/\mathcal{O}) \times \mathcal{D}(Q/\mathcal{O})$ une relation qui témoigne de ce fait. Afin de montrer que $P \simeq_{\mathcal{O}} Q$, nous utilisons la relation $R = \{(P', Q') \mid (P'/\mathcal{O}, Q'/\mathcal{O}) \in R'\}$. D'abord, nous constatons facilement que $(P, Q) \in R$. Ensuite, considérons $(P_1, Q_1) \in R$ (d'où $(P_1/\mathcal{O}, Q_1/\mathcal{O}) \in R'$) avec $P_1 \xrightarrow{\alpha} P_2$ (la preuve pour le cas où $Q_1 \xrightarrow{\alpha} Q_2$ est analogue). Considérons l'action $\beta \in \mathcal{O}(\alpha)$, c'est-à-dire

$$\beta = \begin{cases} \alpha & \text{si } \alpha \in \mathcal{O}(\alpha) \\ \tau & \text{si } \epsilon \in \mathcal{O}(\alpha). \end{cases}$$

D'où $\beta \in \mathcal{O}(\alpha)$. Alors, par la règle `Observation`, nous avons $P_1/\mathcal{O} \xrightarrow{\beta} P_2/\mathcal{O}$. Mais puisque $(P_1/\mathcal{O}, Q_1/\mathcal{O}) \in R'$, il exsite un certain Q_2 tel que $Q_1/\mathcal{O} \xrightarrow{\beta} Q_2/\mathcal{O}$ et $(P_2/\mathcal{O}, Q_2/\mathcal{O}) \in R'$. D'où, par la règle de `Observation`, il existe une suite d'actions γ telle que $\beta \in \mathcal{O}(\gamma)$ et $Q_1 \xrightarrow{\gamma} Q_2$. De plus, nous avons $(P_2, Q_2) \in R'$ car $(P_2/\mathcal{O}, Q_2/\mathcal{O}) \in R'$, et $\mathcal{O}(\alpha) = \mathcal{O}(\beta) = \mathcal{O}(\gamma)$ par la Proposition 3.11. La relation R est donc une $\mathcal{O}$-bisimulation entre P et Q. □

La prochaine proposition affirme que la notion de $\mathcal{O}$-bisimulation faible correspond à la $\mathcal{O}$-bisimulation. Nous pouvons donc conclure que l'équivalence $\mathcal{O}$-bisimulation réduit la relation de bisimulation à celle de la bisimulation faible.

Proposition 3.21. *Soient P et Q des processus et soit $\mathcal{O}$ un critère d'observation. Alors, $P \simeq_{\mathcal{O}} Q$ si et seulement si $P/\mathcal{O} \simeq_{\text{faible}} Q/\mathcal{O}$. En particulier, nous avons*

$$P/\mathcal{O} \simeq Q/\mathcal{O} \quad \textit{si et seulement si} \quad P/\mathcal{O} \simeq_{\text{faible}} Q/\mathcal{O}.$$

Démonstration. Il suffit de démontrer que toute bisimulation faible $P/\mathcal{O} \simeq_{\text{faible}} Q/\mathcal{O}$ est nécessairement une bisimulation forte $P/\mathcal{O} \simeq Q/\mathcal{O}$ (la preuve de l'énoncé réciproque est triviale). Soit R une bisimulation faible entre les processus $P/\mathcal{O}$ et $Q/\mathcal{O}$. D'abord, nous remarquons que $(P/\mathcal{O}, Q/\mathcal{O}) \in R$. Ensuite, considérons $(P_1/\mathcal{O}, Q_1/\mathcal{O}) \in R$ et supposons que $P_1/\mathcal{O} \xrightarrow{\alpha} P_2/\mathcal{O}$ (la preuve pour le cas où $Q_1/\mathcal{O} \xrightarrow{\alpha} Q_2/\mathcal{O}$ est analogue). Puisque R est une bisimulation faible, il existe un calcul $Q_1/\mathcal{O} \stackrel{\alpha}{\Longrightarrow} Q_2/\mathcal{O}$ avec $(P_2/\mathcal{O}, Q_2/\mathcal{O}) \in R$. D'où $Q_1/\mathcal{O} \stackrel{\tau^n\alpha\tau^m}{\longrightarrow} Q_2/\mathcal{O}$ pour certains $n, m \geq 0$. Par la règle `Observation`, il existe un calcul $Q_1 \xrightarrow{\gamma} Q_2$ avec $\tau^n\alpha\tau^m \in \mathcal{O}(\gamma)$, donc $\alpha \in \mathcal{O}(\gamma)$. Ainsi, par la règle `Observation`, il y a une transition $Q_1/\mathcal{O} \xrightarrow{\alpha} Q_2/\mathcal{O}$ avec $(P_2/\mathcal{O}, Q_2/\mathcal{O}) \in R$. La relation R est donc une bisimulation forte entre $P/\mathcal{O}$ et $Q/\mathcal{O}$. □

Exemple 3.22. Considérons les processus P_1 et P_2, dont la sémantique est illustrée à la Figure 3.6. Considérons aussi l'ensemble d'actions $L = \{\beta_1, \beta_2\}$. De la Figure 3.6, nous pouvons déduire que les processus P_1 et P_2 sont $\mathcal{O}_L$-bisimilaires.

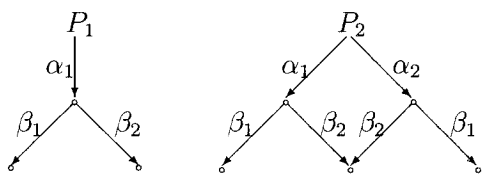

FIG. 3.6 – Illustration de la $\mathcal{O}$-bisimulation.

3.5 Discussion

L'algèbre de processus SPPA est une extension de CCS avec passage de paramètres spécialement conçue pour la modélisation de protocoles de sécurité. Les principaux avantages de cette nouvelle algèbre de processus sont la modélisation, à l'aide du concept d'appel de fonction, des manipulations cryptographiques effec-

tuées par les participants et une modélisation plus détaillée, à l'aide du concept d'action de marquage, des communications entre les participants.

Au Chapitre 4, nous introduisons une extension symbolique de SPPA grâce à laquelle nous obtenons une meilleure formalisation des valeurs symboliques nécessaires aux protocoles de sécurité, telles les nonces et les clés fraîches. En particulier, cette extension devra intégrer une notion de bisimulation symbolique entre les processus.

Au Chapitre 5, SPPA est utilisée dans le contexte d'une nouvelle méthode de vérification basée sur l'*équivalence-checking*. Nous introduisons une formalisation de l'interférence admissible basée sur la $\mathcal{O}$-bisimulation. Ainsi, cette relation d'équivalence et SPPA nous permettent d'établir une caractérisation algébrique pour cette propriété de flots d'information, de même que certaines propriétés de compositionalité par rapport aux principaux opérateurs de SPPA.

CHAPITRE 4

MODÈLE DE SPÉCIFICATION SYMBOLIQUE

Comme nous l'avons remarqué précédemment, l'utilisation des processus avec passage de paramètres combinée à des domaines de messages infinis engendre des graphes de transitions qui ne sont pas à branchement fini, et sur lesquels l'équivalence de bisimulation n'est pas décidable. En effet, une telle sémantique opérationnelle doit prendre en considération tous les messages possibles (voir les règles `Entrée` et `Générateur` de la figure 3.2). L'objectif principal de ce chapitre est d'introduire une extension de SPPA dans laquelle les valeurs symboliques sont spécifiées selon une approche symbolique. Nous considérons un concept abstrait de processus symbolique muni d'une sémantique opérationnelle symbolique capable de manier des absractions de messages, c'est-à-dire des variables symboliques qui n'ont pas de valeurs spécifiques mais qui satisfont certaines contraintes. Une telle extension symbolique de SPPA devra garantir une sémantique finie à tout processus, même si le processus possède des valeurs symboliques qui parcourent des domaines infinis. De plus, ce modèle nous permet d'analyser les effets sur les flots d'information d'un protocole d'un intrus qui y introduit des messages forgés (vus comme valeurs symboliques). En outre, comparativement aux algèbres de processus qui utilisent un système d'inférence pour les manipulations cryptographiques, comme CSPA (voir Section 2.4.1) et le spi calcul (voir Section 2.3.5), SPPA est plus appropriée à l'analyse d'attaques restreintes basées sur une utilisation répétées de messages forgés. En effet, nous constaterons au Chapitre 6 que la modélisation des attaques en SPPA nous permet de restreindre les manipulations cryptographiques qu'un intrus peut effectuer, donc lui imposer l'utilisation de messages aléatoires simples (non cryptés). Ceci n'est pas possible dans un modèle basé sur un système d'inférence qui doit prendre en considération tous les messages que l'intrus peut générer.

Notre modèle de spécification symbolique est basé sur un nouveau concept de *processus contraint*. Un processus contraint correspond à un couple $\langle P, \phi \rangle$ composé d'un processus SPPA P et d'une formule ϕ exprimant un énoncé sur les valeurs symboliques présentes dans P. Le processus P peut donc contenir des variables libres afin de représenter ces valeurs symboliques. De plus, la formule ϕ est tirée d'une logique basée sur l'algèbre de messages présentée à la Section 3.1. Le rôle de cette formule dans le processus contraint consiste à lier les variables libres qui apparaissent lors du déroulement du processus. Par exemple, à un processus qui génère une clé fraîche et qui alloue une variable x pour conserver cette clé, nous lui assignons la formule $\phi ::= \mathcal{K}(x)$ qui affirme que la variable x correspond à une clé. La sémantique opérationnelle d'un processus contraint est donc établie par rapport au comportement de son processus soumis aux restrictions imposées par sa formule. D'où, un processus dont la définition lui permet d'exécuter une action afin d'évoluer vers un autre processus pourra s'exécuter seulement si cette transition satisfait la formule imposée à ce point. Intuitivement, la formule à l'intérieur d'un processus contraint désigne l'ensemble des messages qui peuvent remplacer les variables libres du processus ; cet ensemble de toutes les valeurs possibles évolu en même temps que le processus, par l'ajout de nouvelles variables libres ou de restrictions sur celles qui sont déjà présentes.

Bien que ce chapitre vise l'élaboration d'un modèle similaire à celui de Hennessy-Lin (voir Section 2.5), nous utilisons le concept de processus contraint dans lequel la description des valeurs symboliques est située dans les états plutôt que sur les transitions. En fait, les graphes de transitions symboliques correspondant aux processus contraints s'obtiennent des graphes de transitions de Hennessy-Lin en parcourant tous les chemins possibles. De plus, notre approche, comparativement à celle de Hennessy-Lin, bénéficie d'une logique de messages expressive et dans laquelle nous pouvons définir des énoncés cryptographiques. D'autre part, nous sommes d'avis que le concept de processus contraint est plus pratique pour l'analyse de protocole de sécurité que les graphes de transitions symbolique de Hennessy-Lin. En effet, un processus contraint nous offre un aperçu rapide des valeurs symboliques présentes

à un certain état d'un protocole, ce qui nous épargne la tâche de parcourir tous les chemins menant à cet état.

Notons que le contenu de ce chapitre fut présenté dans [70].

4.1 Logique pour les messages

Nous considérons une logique dont les termes atomiques sont ceux de notre algèbre de messages (voir Section 3.1) et qui possède les *prédicats* suivants :

$$\begin{array}{rcl} \mathcal{P} & ::= & \mathcal{M}(t) \quad (\textit{prédicat de messages}) \quad | \quad \mathcal{N}(t) \quad (\textit{prédicat de nombres}) \\ & | & \mathcal{I}(t) \quad (\textit{prédicat d'identificateurs}) \quad | \quad \mathcal{K}(t) \quad (\textit{prédicat de clés}). \end{array}$$

Les *formules* de notre logique sont construites comme suit :

$$\begin{array}{rcl} \phi & ::= & \mathbf{0} \quad (\textit{faux}) \quad | \quad \mathbf{1} \quad (\textit{vrai}) \\ & | & t == t \quad (\textit{équation de termes}) \quad | \quad \mathcal{P} \quad (\textit{prédicat}) \\ & | & \phi \wedge \phi \quad (\textit{conjonction}) \quad | \quad \exists_x \ \phi \quad (\textit{quantificateur existentiel}). \end{array}$$

L'ensemble des variables libres de la formule ϕ est désigné par $\mathrm{fv}(\phi)$. Étant donnée une formule ϕ et une variable $x \in \mathrm{fv}(\phi)$, $\phi[t/x]$ désigne la formule obtenue de ϕ en substituant chaque occurrence libre de la variable x par le terme $t \in \mathcal{T}$.

Définition 4.1. La formule ϕ est dite *close* si $\mathrm{fv}(\phi) = \emptyset$.

La satisfaction d'une formule close ϕ, notée par $\models \phi$, est définie inductivement de la façon suivante :

- $\models \mathbf{1}$ et $\not\models \mathbf{0}$
- $\models a == b$ si et seulement si les messages a et b sont syntaxiquement identiques, c' est-à-dire
  - $\models n == n$ pour tout $n \in \mathcal{N}$,
  - $\models id == id$ pour tout $id \in \mathcal{I}$,
  - $\models (a_1, a_2) == (b_1, b_2)$ si et seulement si $\models (a_1 == b_1) \wedge (a_2 == b_2)$,
  - $\models \{a_2\}_{a_1} == \{b_2\}_{b_1}$ si et seulement si $\models (a_1 == b_1) \wedge (a_2 == b_2)$,
  - $\models [a_2]_{a_1} == [b_2]_{b_1}$ si et seulement si $\models (a_1 == b_1) \wedge (a_2 == b_2)$, et

- $\models h(a) == h(b)$ si et seulement si $\models a == b$;
- $\models \mathcal{M}(a)$ si et seulement si $a \in \mathcal{M}$;
- $\models \mathcal{N}(a)$ si et seulement si $a \in \mathcal{N}$;
- $\models \mathcal{I}(a)$ si et seulement si $a \in \mathcal{I}$;
- $\models \mathcal{K}(a)$ si et seulement si $a \in \mathcal{K}$;
- $\models \phi \wedge \phi'$ si et seulement si $\models \phi$ et $\models \phi'$;
- $\models \exists_x \phi$ si et seulement si $\models \phi[a/x]$ pour un certain $a \in \mathcal{M}$.

Nous supposons que les prédicats $\mathcal{N}$, $\mathcal{I}$ et $\mathcal{K}$ sont décidables[1].

Définition 4.2. Soit ϕ une formule telle que $\text{fv}(\phi) = \{x_1, \ldots, x_n\}$. Étant donnée une évaluation $\varrho : \mathcal{V} \to \mathcal{M}$, $\varrho(\phi)$ désigne la formule close $\phi[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$.

La satisfaction de la formule ϕ par l'évaluation ϱ, notée par $\varrho \models \phi$, est définie comme suit :

$$\varrho \models \phi \quad \text{si et seulement si} \quad \models \varrho(\phi)$$

Définition 4.3. Les formules ϕ et ϕ' sont *équivalentes* – noté par $\phi \Leftrightarrow \phi'$ – si

$$\varrho \models \phi \quad \text{si et seulement si} \quad \varrho \models \phi'$$

pour toute évaluation ϱ.[2]

Exemple 4.4. Considérons la formule

$$\phi ::= \exists_{x_1} \exists_{x_2} \, (x == (x_1, x_2) \wedge \mathcal{K}(x_1) \wedge \mathcal{M}(x_2))$$

avec $\text{fv}(\phi) = \{x\}$. Cette formule affirme que la variable x correspond à un couple composé d'une clé et d'un message. Donc, si l'évaluation ϱ_1 est telle que $\varrho_1(x) = (k, a)$, avec $k \in \mathcal{K}$ et $a \in \mathcal{M}$, alors nous pouvons constater que $\varrho_1 \models \phi$. Cependant, si nous considèrons une évaluation ϱ_2 telle que $\varrho_2(x) = (a, a)$, alors $\varrho_2 \not\models \phi$ à

[1] En outre, ils ne sont jamais satisfaits par un message non atomique (rappelons que nous avons supposé plus haut que $\mathcal{K} = \mathcal{N}$). Par exemple, $\not\models \mathcal{I}(h(a))$ et $\not\models \mathcal{N}(\{a\}_b)$ pour tout $a, b \in \mathcal{M}$.

[2] En particulier, une formule ϕ est équivalente à $\mathbf{0}$ – noté par $\phi \Leftrightarrow \mathbf{0}$ – si $\varrho \not\models \phi$ pour toute évaluation ϱ.

moins que $a \in \mathcal{K}$. D'autre part, si nous considèrons une évaluation ϱ_3 telle que $\varrho_2(x) = \{a\}_k$, alors $\varrho_3 \not\models \phi$.

Lemme 4.5. *Toute équation $t == t'$ (avec $t, t' \in \mathcal{T}$) est équivalente à une conjonction finie d'équations irréductibles $x == t''$ (avec $x \in \mathcal{V}$ et $t'' \in \mathcal{T}$).*

Démonstration. Ce résultat se déduit directement de la définition inductive de la satisfaction $\models (a == b)$ donnée plus haut. □

4.1.1 Formules caractéristiques

À l'aide de notre logique, nous pouvons assigner à chaque fonction $f \in \mathcal{F}$ une formule qui caractérise son domaine. De façon analogue, nous assignons à toute fonction génératrice $new \in \mathcal{F}$ une formule qui caractérise son image.

Définition 4.6. 1. La *formule caractéristique* de la fonction $f(x_1, \ldots, x_n)$ est donnée par la formule $\phi_{f(x_1,\ldots,x_n)}$ (ou simplement ϕ_f) telle que $\mathrm{fv}(\phi_f) = \{x_1, \ldots, x_n\}$ et

$$\models \phi_f[a_1/x_1] \ldots [a_n/x_n] \quad \text{si et seulement si} \quad (a_1, \ldots, a_n) \in \mathrm{dom}(f).$$

2. La *formule caractéristique* de la fonction génératrice $new(-)$ est donnée par la formule ϕ_{new} telle que $\mathrm{fv}(\phi_{new}) = \{x\}$ et

$$\models \phi_{new}[a/x] \quad \text{si et seulement si} \quad a \in \mathrm{im}(new).$$

Nous écrivons souvent $\phi_f(a_1, \ldots, a_n)$ à la place de $\phi_f[a_1/x_1] \ldots [a_n/x_n]$, ou simplement $\phi_f(a)$ avec $a = (a_1, \ldots, a_n)$. Dans le cas d'une fonction génératrice $new \in \mathcal{F}$, nous écrivons $\phi_{new}(a)$ à la place de $\phi_{new}[a/x]$.

4.1.2 Décidabilité

L'objectif de cette section est de démontrer que la logique définie ci-dessus est décidable. Ce résultat provient du fait qu'elle est restreinte aux opérateurs de

conjonction et de quantification existentielle. Afin de prouver ce résultat, nous devons d'abord montrer que toute formule close est équivalente à une formule (close) sans quantificateur. Par conséquent, la décidabilité d'une formule se ramène à la satisfaction d'un nombre fini de prédicats et d'équations, qui sont tous décidables par hypothèse.

Lemme 4.7. *Toute formule close ϕ est équivalente à une formule de la forme*

$$\exists_{x_1} \ldots \exists_{x_n} \ (\phi_1 \wedge \ldots \wedge \phi_m)$$

où les sous-formules ϕ_i sont des prédicats ou des équations de la forme $x == t$, avec $x \in \{x_1, \ldots, x_n\}$ et $\mathrm{fv}(t) \subseteq \{x_1, \ldots, x_n\}$.

Démonstration. Étant donnée une formule close ϕ, nous pouvons toujours supposer qu'elle est donnée dans sa forme normale[3] :

$$\phi ::= \ \exists_{x_1} \ldots \exists_{x_n} \ (\phi_1 \wedge \ldots \wedge \phi_m)^4$$

où les ϕ_i sont sans quantificateur.[5] Par le lemme 4.5, toute sous-formule ϕ_i est équivalente à une conjonction finie d'équations irréductibles $x == t$. □

Considérons maintenant la famille $\mathcal{E}$ de formules closes définie par :

$$\begin{aligned}\mathcal{E} \ = \ \{ \ & \exists_{x_1} \ldots \exists_{x_n} \ (\phi_1 \wedge \ldots \wedge \phi_m) \mid \phi_i ::= \mathcal{K}(t) \text{ ou } \phi_i ::= \mathcal{I}(t) \text{ ou } \phi_i ::= \mathcal{N}(t) \\ & \text{ou } \phi_i ::= \mathcal{M}(t) \text{ ou } \phi_i ::= x == t, \text{ pour certains } n, m \in \mathbb{N}, \\ & \text{pour un certain } x \in \{x_1, \ldots, x_n\} \text{ et pour un certain } t \text{ tel que} \\ & \mathrm{fv}(t) \subseteq \{x_1, \ldots, x_n\} \ \} \ \cup \ \{\mathbf{0}, \mathbf{1}\}\end{aligned}$$

Ainsi, par le Lemme 4.7, toute formule close ϕ est équivalente à une formule $\phi' \in \mathcal{E}$.

---

[3]Nous supposons que les formules **1** et **0** coïncident avec leur forme normale.

[4]Si $\phi ::= (\exists_x \ \phi_1) \wedge \phi_2$ alors ϕ est équivalente à la formule $\exists_y \ (\phi_1[y/x] \wedge \phi_2)$, où y est une variable qui n'apparaît pas dans ϕ_2.

[5]D'où $\phi_i ::= \ \mathcal{K}(t)$, $\mathcal{I}(t)$, $\mathcal{N}(t)$ ou $\mathcal{M}(t)$, ou $\phi_i ::= \ t == t'$, avec $\mathrm{fv}(t), \mathrm{fv}(t') \subseteq \{x_1, \ldots, x_n\}$.

Lemme 4.8. *Toute formule $\phi \in \mathcal{E}$ est équivalente à une formule sans quantificateur $\psi \in \mathcal{E}$.*

Démonstration. Nous procédons par induction sur le nombre de quantificateurs existentiels dans ϕ. Le cas où ϕ ne possède aucun quantificateur est trivial.

Soit $n \geq 0$ et supposons que toute formule dans $\mathcal{E}$ ayant au plus n quantificateurs existentiels est équivalente à une certaine formule sans quantificateur appartenant à la famille $\mathcal{E}$. Considérons une formule quelconque $\phi \in \mathcal{E}$, avec

$$\phi \ ::= \ \exists_x \exists_{x_1} \ldots \exists_{x_n} \ (\phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_m).$$

D'abord, nous pouvons supposer qu'aucune sous-formule ϕ_i est un prédicat $\mathcal{M}(t)$; sinon, nous pouvons les enlever et obtenir une formule équivalente. Supposons maintenant que $\phi_i \ ::= \ x == t$ pour un certain i (disons $i = 1$). Si $t = x$ (c'est-à-dire $\phi_1 \ ::= \ x == x$), alors nous pouvons retirer ϕ_1 et supposer que $\phi \ ::= \ \exists_x \exists_{x_1} \ldots \exists_{x_n} \ (\phi_2 \wedge \ldots \wedge \phi_m)$. Sinon, si x apparaît dans t, alors ϕ est équivalente à $\mathbf{0}$, donc $\not\models \phi$, car nous n'acceptons pas les messages infinis. Réciproquement, si x n'apparaît pas dans t, alors nous constatons que ϕ est équivalente à la formule

$$\exists_{x_1} \ldots \exists_{x_n} \ (\phi_2[t/x] \wedge \ldots \wedge \phi_m[t/x])$$

qui possède un quantificateur de moins que ϕ. Or, puisque cette formule appartient à $\mathcal{E}$, la preuve se complète en utilisant l'hypothèse d'induction qui nous assure l'existence d'une formule sans quantificateur $\phi' \in \mathcal{E}$ équivalente à la formule ci-dessus, donc équivalente à ϕ. De plus, nous voyons que toute équation présente dans ϕ peut être retirée de cette façon : toute sous-formule $\phi_i \ ::= \ t == t'$ est remplacée par une conjonction d'équations équivalentes $x'_1 == t_1 \wedge \ldots \wedge x'_k == t_k$, et il suffit de répéter la démarche présentée ci-dessus pour chaque variable x'_j.

Supposons maintenant que la formule ϕ ne possède aucune équation parmi ses sous-formules ϕ_i. Supposons aussi que la variable x apparaît seulement dans les sous-formules $\phi_1, \ldots, \phi_k$ (pour $k \leq m$). Puisque les formules $\exists_x \mathcal{I}(t)$, $\exists_x \mathcal{N}(t)$

et $\exists_x \mathcal{K}(t)$ sont satisfaites seulement lorsque $t = x$, aucune autre des variables quantifiées $x_1, \ldots, x_m$ ne peut apparaître dans les prédicats $\phi_1, \ldots, \phi_k$ (sinon $\not\models \phi$). Par conséquent, la formule ϕ est équivalente à

$$\exists_x(\phi_1 \wedge \ldots \wedge \phi_k) \ \wedge \ \exists_{x_1} \ldots \exists_{x_n} \ (\phi_{k+1} \wedge \ldots \wedge \phi_m)$$

où $\phi_i \in \{\mathcal{K}(x), \mathcal{I}(x), \mathcal{N}(x)\}$ (pour $1 \leq i \leq k$). D'où, il suffit de trouver une formule sans quantificateur ψ équivalente à $\exists_x(\phi_1 \wedge \ldots \wedge \phi_k)$; nous prenons $\psi ::= \mathbf{1}$ si

- $\phi_i ::= \mathcal{I}(x)$, pour tout $i = 1, \ldots, k$; ou
- $\phi_i ::= \mathcal{N}(x)$ ou $\phi_i ::= \mathcal{K}(x)$, pour tout $i = 1, \ldots, k$.

Sinon, nous prenons $\psi ::= \mathbf{0}$. Finalement, il est facile de voir que la formule résultante (soit $\mathbf{0}$, soit $\exists_{x_1} \ldots \exists_{x_n} \ (\phi_{k+1} \wedge \ldots \wedge \phi_m)$) appartient à la famille $\mathcal{E}$ et possède au plus n quantificateurs. Ainsi, par l'hypothèse d'induction, nous pouvons lui trouver une formule équivalente $\phi' \in \mathcal{E}$, qui est également équivalente à ϕ. □

Pour le lemme suivant, rappelons que toute formule dans $\mathcal{E}$ est close, y compris les formules sans quantificateur.

Lemme 4.9. *Toute formule sans quantificateur dans $\mathcal{E}$ est décidable.*

Démonstration. Soit $\phi \in \mathcal{E}$ une formule sans quantificateur. Si $\phi ::= \mathbf{1}$ ou $\phi ::= \mathbf{0}$, alors l'énoncé est trivial. Supposons maintenant que $\phi ::= \ \phi_1 \wedge \ldots \wedge \phi_n$ avec $n \geq 1$. Puisque ϕ est close, toute sous-formule ϕ_i est soit un prédicat $\mathcal{I}(a)$, soit $\mathcal{K}(a)$, soit $\mathcal{N}(a)$ (tout prédicat $\mathcal{M}(a)$ est immédiatement remplacé par $\mathbf{1}$), soit une équation $a == a'$, pour certains messages $a, a' \in \mathcal{M}$. Puisque les prédicats $\mathcal{N}$, $\mathcal{I}$ et $\mathcal{K}$ sont décidables et que toute équation $a == a'$ est aussi décidable par applications successives d'un nombre fini de réductions, nous pouvons conclure que toute sous-formule ϕ_i peut être individuellement remplacée soit par $\mathbf{1}$, soit par $\mathbf{0}$. Une telle conjonction de $\mathbf{1}$ et de $\mathbf{0}$ est clairement décidable. □

Théorème 4.10. *Toute formule de notre logique est décidable.*

Démonstration. Étant donnée une formule close ϕ, par le Lemme 4.7, nous pouvons trouver une formule (close) équivalente $\phi' \in \mathcal{E}$. Par le Lemme 4.8, ϕ' est à son

tour équivalente à une formule sans quantificateur $\psi \in \mathcal{E}$, qui est décidable par le Lemme 4.9. D'où ϕ est décidable. □

4.2 Relations entre sous-ensembles finis de variables

Dans cette section, nous introduisons la notion de relation finie entre variables, que nous utiliserons afin de lier les variables libres des processus contraints dans le but de les comparer.

Considérons la famille $\mathfrak{R}$ de toutes les relations entre des sous-ensembles finis de variables, d'où

$$\mathfrak{R} = \{R \mid R \subseteq \mathcal{V}_1 \times \mathcal{V}_2 \text{ pour certains sous-ensembles finis } \mathcal{V}_1, \mathcal{V}_2 \subseteq \mathcal{V}\}.$$

Définition 4.11. La relation $R \in \mathfrak{R}$ est une *relation pleine entre* $\mathcal{V}_1$ *et* $\mathcal{V}_2$ si

$$\forall_{x \in \mathcal{V}_1} \exists_{y \in \mathcal{V}_2} \ (x, y) \in R \quad \text{et} \quad \forall_{y \in \mathcal{V}_2} \exists_{x \in \mathcal{V}_1} \ (x, y) \in R.$$

Dans le cas où $\mathcal{V}_1 = \emptyset$ ou $\mathcal{V}_2 = \emptyset$, alors nous considérons la relation vide $R = \emptyset$ comme étant pleine.

La prochaine définition introduit des opérateurs de substitution sur les relations de $\mathfrak{R}$.

Définition 4.12. Soient $x, y \in \mathcal{V}$ des variables et soit $R \in \mathfrak{R}$ une relation.

1. Nous désignons par $R[x]$ la relation de $\mathfrak{R}$ définie par

$$R[x] = \{(x', y') \in R \mid x' \neq x \text{ et } y' \neq x\}.$$

2. Nous désignons par $R[\![(x, y)]\!]$ la relation de $\mathfrak{R}$ définie par

$$R[\![(x, y)]\!] = R[x][y] \ \cup \ \{(x, y)\}.$$

3. Étant donnée une évaluation ϱ, nous désignons par $\varrho[x/y]$ l'évaluation définie par

$$\varrho[x/y](z) = \begin{cases} \varrho(x) & \text{si } z = y \\ \varrho(z) & \text{sinon.} \end{cases}$$

La relation $R[\![(x,y)]\!]$ s'obtient donc de R en retirant, tout d'abord, toute occurrence de x et y, puis en ajoutant le couple (x,y).

Définition 4.13. L'évaluation ϱ est *cohérente* avec la relation $R \in \mathfrak{R}$ si $\varrho(x) = \varrho(y)$ dès que $(x,y) \in R$.

Le prochain lemme nous permet d'affirmer que cet opérateur de substitution pour les évaluations respecte la relation de cohérence par rapport à une relation, modulo une substitution analogue sur cette dernière. Sa preuve se déduit facilement des Définitions 4.12 et 4.13.

Lemme 4.14. *Si l'évaluation ϱ est cohérente avec la relation $R[x]$, alors $\varrho[x/y]$ est cohérente avec $R[\![(x,y)]\!]$.*

4.2.1 Relation d'équivalence sur les évaluations

Dans cette section, nous introduisons une relation d'équivalence sur les évaluations qui est définie de façon à satisfaire trois conditions essentielles à la méthode de preuve présentée à la Section 4.7. Tout d'abord, elle ne doit posséder qu'un nombre fini de classes d'équivalences. Elle doit aussi être préservée par rapport à l'opérateur de substitution de variables (Définition 4.12). Finalement, elle doit respecter les propriétés de cohérence par rapport à toute relation (Définition 4.13).

Pour les besoins de cette section, nous considérons les deux ensembles de formules suivants :

$$E_1 = \{\phi_1, \ldots, \phi_m\} \quad \text{et} \quad E_2 = \{\psi_1, \ldots, \psi_{m'}\}$$

avec

$$\bigcup_{1\leq j\leq m} \mathrm{fv}(\phi_j) = \{x_1,\ldots,x_n\} \quad \text{et} \quad \bigcup_{1\leq j\leq m'} \mathrm{fv}(\psi_j) = \{y_1,\ldots,y_{n'}\}.$$

Définition 4.15. Les évaluations ϱ et ϱ' sont *équivalentes par rapport aux ensembles* E_1 *et* E_2, noté par $\varrho \equiv_{E_1E_2} \varrho'$, si

- pour tout $1 \leq i \leq m$, $\varrho \models \phi_i$ si et seulement si $\varrho' \models \phi_i$;
- pour tout $1 \leq i \leq m'$, $\varrho \models \psi_i$ si et seulement si $\varrho' \models \psi_i$;
- pour tout $z, z' \in \{x_1,\ldots,x_n\} \cup \{y_1,\ldots,y_{n'}\}$,

$$\varrho(z) = \varrho(z') \quad \text{si et seulement si} \quad \varrho'(z) = \varrho'(z').$$

La classe d'équivalence d'une évaluation ϱ par rapport aux ensembles E_1 et E_2 est définie par :

$$[\![\varrho]\!]_{E_1E_2} = \{\varrho' \mid \varrho' \equiv_{E_1E_2} \varrho\}.$$

Lemme 4.16. *Le nombre de classes d'équivalence* $[\![\varrho]\!]_{E_1E_2}$ *est fini, c'est-à-dire l'ensemble* $\{[\![\varrho]\!]_{E_1E_2} \mid \varrho$ *est une évaluation*$\}$ *est fini.*

Démonstration. Étant donnés les ensembles finis E_1 et E_2 donnés plus haut, nous pouvons constater que

- il y a au plus 2^m classes d'équivalence pour la relation :

$$\forall_{1\leq i\leq m} \quad \varrho \models \phi_i \quad \text{si et seulement si} \quad \varrho' \models \phi_i;$$

- il y a au plus $2^{m'}$ classes d'équivalence pour la relation :

$$\forall_{1\leq i\leq m'} \quad \varrho \models \psi_i \quad \text{si et seulement si} \quad \varrho' \models \psi_i;$$

- il y a au plus $2^{(n+n')^2}$ classes d'équivalence pour la relation :

$$\forall_{z,z'\in\{x_1,\ldots,x_n\}\cup\{y_1,\ldots,y_{n'}\}} \quad \varrho(z) = \varrho(z') \text{ si et seulement si } \varrho'(z) = \varrho'(z').$$

Ainsi, le nombre de classes d'équivalence $[\![\varrho]\!]_{E_1E_2}$ est borné par $2^{m+m'+(n+n')^2}$. □

Lemme 4.17. *Soient ϱ et ϱ' des évaluations telles que $\varrho \equiv_{E_1E_2} \varrho'$.*

1. *Pour toute relation $R \subseteq \{x_1, \ldots, x_n\} \times \{y_1, \ldots, y_{n'}\}$[6], ϱ est cohérente avec R si et seulement si ϱ' est cohérente avec R.*
2. *Pour toutes variables $x, y \in \{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_{n'}\}$,*

$$\varrho[x/y] \equiv_{E_1E_2} \varrho'[x/y] \quad \textit{et} \quad \varrho[y/x] \equiv_{E_1E_2} \varrho'[y/x].$$

Les preuves des énoncés du Lemme 4.17 s'obtiennent directement de la définition de la relation d'équivalence $\equiv_{E_1E_2}$.

4.3 Processus contraints

Dans cette section, nous introduisons un modèle qui peut s'appliquer à plusieurs algèbres de processus. En effet, étant donnée une algèbre de processus, nous commençons par étendre sa syntaxe de façon à formaliser la génération de valeurs symboliques en terme d'un opérateur de séquentialité (e.g. les appels de fonction « `let` $x = new(-)$ `in` P »). Les messages ainsi générés sont ensuite typés à l'aide d'une formule caractéristique ϕ_{new}. Par souci de simplicité, nous exposons notre modèle symbolique dans le contexte de l'algèbre de processus SPPA.

Hypothèses. Pour les besoins du modèle symbolique présenté dans ce chapitre, nous établissons les restrictions suivantes sur les processus SPPA. Tout d'abord, nous ne prenons pas en considération les actions d'échec de SPPA ($\mathtt{fail}^{\mathtt{f}}_{\mathtt{id}}$, $\mathtt{fail}^{\mathtt{dec}}_{\mathtt{id}}$, $\mathtt{fail}^{\mathtt{signv}}_{\mathtt{id}}$ et $\mathtt{fail}^{=}_{\mathtt{id}}$). Ces actions sont exclues puisqu'elles nécessitent l'introduction d'un opérateur de négation dans notre logique, et nous ne savons pas si cet ajout préserve la décidabilité de la logique. Pour ces mêmes raisons, nous devons restreindre l'opérateur de restriction $P \setminus L$ aux sous-ensembles d'actions L tels que

[6] *avec $R \in \mathfrak{R}$.*

son complément $Act \setminus L$ est définissable dans notre logique, c'est-à-dire il existe une formule ϕ_α^L telle que $\mathrm{fv}(\phi_\alpha^L) = \mathrm{fv}(\alpha)$ et, pour toute évaluation ϱ,

$$\varrho \models \phi_\alpha^L \quad \text{si et seulement si} \quad \varrho(\alpha) \in L.$$

Définition 4.18. Les formules ϕ et ϕ' sont *$\mathcal{V}$-équivalentes*, noté par $\phi \overset{\mathcal{V}}{\Leftrightarrow} \phi'$, si

$$\phi \overset{\mathcal{V}}{\Leftrightarrow} \phi' \quad \text{si et seulement si} \quad \phi \Leftrightarrow \phi' \text{ et } \mathrm{fv}(\phi) = \mathrm{fv}(\phi').$$

La classe d'équivalence d'une formule ϕ sous la relation $\overset{\mathcal{V}}{\Leftrightarrow}$ est définie par

$$[\![\phi]\!]_{\mathcal{V}} = \{\phi' \mid \phi \overset{\mathcal{V}}{\Leftrightarrow} \phi'\}.$$

Définition 4.19. Le *processus contraint* obtenu du processus P et de la formule ϕ, noté par $\langle P,\ \phi\rangle$, est défini par l'ensemble

$$\langle P,\ \phi\rangle = \{(P, \phi') \mid \phi \overset{\mathcal{V}}{\Leftrightarrow} \phi'\}.$$

La formule ϕ dans le processus contraint a pour but de restreindre les variables apparaissant dans P. Ainsi, si $\phi \overset{\mathcal{V}}{\Leftrightarrow} \phi'$ (c'est-à-dire les formules ϕ et ϕ' sont équivalentes et ont les mêmes variables libres), alors les processus contraints $\langle P,\ \phi\rangle$ et $\langle P,\ \phi'\rangle$ sont considérés identiques car le processus P est soumis à des contraintes équivalentes.

4.3.1 Spécification symbolique des protocoles

L'algèbre de processus SPPA combinée au concept de processus contraint nous procure un modèle très expressif pour la spécification de protocoles sécurité. L'idée principale derrière ce modèle consiste, tout d'abord, à obtenir des spécifications de chaque participant du protocole en terme de processus contraints, à partir de celles en terme de processus SPPA. Par exemple, un participant A est spécifié par le processus contraint $\langle A,\ \phi_A\rangle$, où $A ::= (S_A, id_A)$ est un participant SPPA et

ϕ_A est une formule caractérisant ses connaissances initiales (formalisées par des variables libres dans l'agent initial S_A). Notons que les connaissances initiales d'un participant (e.g. ses clés privées et les clés publiques des autres participants) sont communément spécifiées implicitement dans l'agent initial en tant que messages $m \in \mathcal{M}$ ou clés $k \in \mathcal{K}$. D'où, un agent initial S_A est habituellement clos[7] et, dans ce cas, nous avons $\phi_A ::= \mathbf{1}$. Cependant, notre modèle symbolique permet une représentation de ces connaissances initiales en tant que valeurs symboliques (c'est-à-dire des variables libres) dans l'agent initial. Par exemple, si $\mathrm{fv}(S_A) = \{x\}$ et x désigne, dans la définition de S_A, une clé privée appartenant à A, alors nous posons $\phi_A ::= \mathcal{K}(x)$.

Étant donnée une spécification de chaque participant d'un protocole, disons $\langle A,\ \phi_A\rangle$, $\langle B,\ \phi_B\rangle$ et $\langle S,\ \phi_S\rangle$, le protocole entier est alors spécifié par le processus contraint $\langle P,\ \phi_P\rangle$, avec $P ::= A \parallel B \parallel S$ et $\phi_P ::= \phi_A \wedge \phi_B \wedge \phi_S$.

De façon analogue, un intrus qui tente d'attaquer le protocole est spécifié indépendamment des autres participants. La spécification d'un intrus est donc donnée par un processus contraint $\langle E,\ \phi_E\rangle$, communément appelé *processus ennemi*, où $E ::= (S_E, id_E)$ est un participant SPPA, S_E désigne l'agent initial de E (c'est-à-dire l'agent SPPA spécifiant l'attaque menée par l'intrus) et ϕ_E est une formule qui caractérise les connaissances initiales de l'intrus (comme plus haut). À partir de cette notation, la spécification du protocole P attaqué par le processus ennemi E est simplement donnée par le processus contraint $\langle P_E,\ \phi_{P_E}\rangle$, avec $P_E ::= P \parallel E$ et $\phi_{P_E} ::= \phi_P \wedge \phi_E$.

Exemple 4.20. Considérons le protocole à une étape suivant :

$$\text{Message 1 :} \quad A \overset{\{n_A\}_{k_B}}{\longrightarrow} B$$

dans lequel le participant A génère d'abord un nonce frais n_A, puis envoie au participant B ce nonce chiffré avec la clé publique de B (désignée par k_B). Les participants A et B sont respectivement spécifiés par les participants SPPA $A =$

[7] c'est-à-dire $\mathrm{fv}(S_A) = \emptyset$.

(S_A, id_A) et $B = (S_B, id_B)$, où id_A est l'identificateur de A, id_B est l'identificateur de B, et les agents initiaux S_A et S_B sont définis comme suit :

$$S_A ::= \texttt{let } x_1 = \text{newNumber}(-) \texttt{ in let } x_2 = \text{enc}(k_B, x_1) \texttt{ in } \overline{c}(x_2).\mathbf{0}$$
$$S_B ::= c(y_1).\ \texttt{case } y_1 \texttt{ of } \{y_2\}_{k_B} \texttt{ in } \mathbf{0}\ .$$

Le protocole est alors spécifié par le processus SPPA $P ::= A \parallel B$, dans lequel les participants A et B peuvent communiquer par l'entremise du canal public c. Puisque les agents S_A et S_B sont clos, les processus contraints correspondant aux participants du protocole sont donnés par $\langle A,\ \mathbf{1}\rangle$ et $\langle B,\ \mathbf{1}\rangle$. Ainsi, la spécification du protocole est donnée par le processus contraint $\langle P,\ \mathbf{1}\rangle$.

Notons que, dans cet exemple, le participant A possède comme connaissance initiale la clé publique k_B. Dans le but d'exprimer A indépendamment de l'autre participant B, nous pouvons remplacer le message k_B dans la définition de l'agent S_A par une variable libre x. Nous obtenons ainsi le participant $A' = (S_{A'}, id_A)$ avec

$$S_{A'} ::= \texttt{let } x_1 = \text{newNumber}(-) \texttt{ in let } x_2 = \text{enc}(x, x_1) \texttt{ in } \overline{c}(x_2).\mathbf{0}$$

et le processus contraint correspondant $\langle A,\ \mathcal{K}(x)\rangle$ qui nous assure que la variable libre x correspond à une clé quelconque. Dans ce cas, la spécification du protocole est donnée par $\langle P',\ \mathcal{K}(x)\rangle$, avec $P' ::= A' \parallel B$.

4.4 Sémantique symbolique

Rappelons que la sémantique opérationnelle avec passage de paramètres de SPPA (voir Section 3.2.4) est seulement définie pour les processus clos.[8] Nous avons aussi remarqué plus haut que ce type de sémantique engendre fréquemment des graphes de transitions infinis. Dans cette section, nous établissons une sémantique opérationnelle symbolique pour les processus contraints, qui leur assigne des graphes de transitions finis.

[8] processus sans variable libre.

La sémantique opérationnelle symbolique des processus contraints est définie dans les Figures 4.1 et 4.2. Elle s'inspire de la sémantique symbolique présentée par Hennessy & Lin [57] où les valeurs booléennes qui servent de gardes aux actions sont remplacées par des formules ϕ restreignant les variables libres du processus. Il est important de noter que nous supposons qu'une transition $\langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle$ est éliminée lorsque ϕ ou ϕ' est équivalente à $\mathbf{0}$.[9] Notons également que l'introduction de prédicats $\mathcal{M}(x)$ dans les formules des processus contraints est essentielle à l'identification des variables libres. En effet, il suffit de remarquer que les processus contraints $\langle P,\ \mathbf{1}\rangle$ et $\langle P,\ \mathcal{M}(x)\rangle$ sont distincts car $\mathbf{1} \overset{\mathcal{V}}{\not\Leftrightarrow} \mathcal{M}(x)$, bien que $\mathbf{1} \Leftrightarrow \mathcal{M}(x)$. D'autre part, cette sémantique symbolique respecte la définition des processus contraints : étant donnés un processus P et des formules $\mathcal{V}$-équivalentes ϕ_1 et ϕ_2, si $\langle P,\ \phi_1\rangle \xrightarrow{\alpha} \langle P,\ \phi_1'\rangle$ et $\langle P,\ \phi_2\rangle \xrightarrow{\alpha} \langle P,\ \phi_2'\rangle$, alors $\phi_1' \overset{\mathcal{V}}{\Leftrightarrow} \phi_2'$, d'où $\langle P,\ \phi_1'\rangle = \langle P,\ \phi_2'\rangle$.

Rappelons que les processus contraints $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ ne peuvent pas avoir des variables en commun (voir Section 3.2.1), d'où $\mathrm{fv}(\phi) \cap \mathrm{fv}(\psi) = \emptyset$. Dans la règle `Restriction`, la formule ϕ_α^L désigne la formule qui affirme que l'action α n'appartient pas à L (voir Section 4.3). Dans la règle `Observation`, le *calcul* $\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle$, pour une suite d'actions $\gamma = \alpha_0\ \ldots \alpha_n \in Act^*$, désigne la suite finie de transitions :

$$\langle P,\ \phi\rangle \xrightarrow{\alpha_0} \cdots \xrightarrow{\alpha_n} \langle P',\ \phi'\rangle.$$

Définition 4.21. Soit $\langle P,\ \phi\rangle$ un processus contraint.

1. Le processus contraint $\langle P',\ \phi'\rangle$ est une *dérivée* de $\langle P,\ \phi\rangle$ s'il existe un calcul $\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle$ pour un certain $\gamma \in Act^*$. L'ensemble des dérivées de $\langle P,\ \phi\rangle$ est donné par

$$\mathcal{D}(\langle P,\ \phi\rangle) = \{\langle P',\ \phi'\rangle \mid \exists_{\gamma \in Act^*}\ \langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle\}.$$

[9] c'est-à-dire $\phi \Leftrightarrow \mathbf{0}$ ou $\phi' \Leftrightarrow \mathbf{0}$.

Sortie

$$\frac{-}{\langle \overline{c}(t).A,\ \phi\rangle \xrightarrow{\overline{c_{id}}_A(t)} \langle A,\ \phi\rangle}$$

Entrée

$$\frac{-}{\langle c(x).A,\ \phi\rangle \xrightarrow{c_{id}{}_A(x)} \langle A,\ (\exists_x \phi) \wedge \mathcal{M}(x)\rangle}$$

Fonction

$$\frac{f \in \mathcal{F}}{\langle \texttt{let } x{=}f(t) \texttt{ in } A,\ \phi\rangle \xrightarrow{f_{id}{}_A} \langle A,\ (\exists_x \phi) \wedge \phi_f(t) \wedge x{=}{=}f(t)\rangle}$$

Générateur

$$\frac{new \in \mathcal{F}}{\langle \texttt{let } x{=}f(-) \texttt{ in } A,\ \phi\rangle \xrightarrow{new_{id}{}_A} \langle A,\ (\exists_x \phi) \wedge \phi_{new}(x)\rangle}$$

Extraction

$$\frac{-}{\langle \texttt{let } (x,y){=}(t,t') \texttt{ in } A,\ \phi\rangle \xrightarrow{\mathrm{split}_{id}{}_A} \langle A,\ (\exists_x \exists_y \phi) \wedge x{=}{=}t \wedge y{=}{=}t'\rangle}$$

Décryption

$$\frac{-}{\langle \texttt{case } t \texttt{ of } \{x\}_{t'} \texttt{ in } A,\ \phi\rangle \xrightarrow{\mathrm{dec}_{id}{}_A} \langle A,\ (\exists_x \phi) \wedge \mathcal{K}(t') \wedge t{=}{=}\{x\}_{t'}\rangle}$$

Signature-Vérif

$$\frac{-}{\langle \texttt{case } t \texttt{ of } [t'']_{t'} \texttt{ in } A,\ \phi\rangle \xrightarrow{\mathrm{signv}_{id}{}_A} \langle A,\ \phi \wedge \mathcal{K}(t') \wedge t{=}{=}[t'']_{t'}\rangle}$$

Comparaison

$$\frac{\langle A,\ \phi\rangle \xrightarrow{\alpha} \langle A',\ \phi'\rangle}{\langle [t{=}t']A,\ \phi\rangle \xrightarrow{\alpha} \langle A',\ \psi\rangle}$$

avec $\psi ::= \begin{cases} (\exists_x(\phi \wedge t{=}{=}t')) \wedge \psi' & \text{si } \alpha{=}\ c_{id}(x),\ f_{id},\ \mathrm{split}_{id} \text{ ou } \mathrm{dec}_{id} \\ & \text{et } \phi' ::= (\exists_x \phi) \wedge \psi' \\ \phi' \wedge t{=}{=}t' & \text{sinon.} \end{cases}$

FIG. 4.1 – Sémantique des processus contraints.

Somme

$$\frac{\langle A,\ \phi\rangle \xrightarrow{\alpha} \langle A',\ \phi'\rangle \quad \text{et} \quad \mathrm{fv}(\phi)\cap\mathrm{fv}(\psi)=\emptyset}{\langle A+B,\ \phi\wedge\psi\rangle \xrightarrow{\alpha} \langle A',\ \phi'\wedge\psi\rangle} \quad \text{et}$$

$$\frac{\langle B,\ \psi\rangle \xrightarrow{\alpha} \langle B',\ \psi'\rangle \quad \text{et} \quad \mathrm{fv}(\phi)\cap\mathrm{fv}(\psi)=\emptyset}{\langle A+B,\ \phi\wedge\psi\rangle \xrightarrow{\alpha} \langle B',\ \phi\wedge\psi'\rangle}$$

Parallèle

$$\frac{\langle A,\ \phi\rangle \xrightarrow{\alpha} \langle A',\ \phi'\rangle \quad \text{et} \quad \mathrm{fv}(\phi)\cap\mathrm{fv}(\psi)=\emptyset}{\langle A|B,\ \phi\wedge\psi\rangle \xrightarrow{\alpha} \langle A'|B,\ \phi'\wedge\psi\rangle} \quad \text{et}$$

$$\frac{\langle B,\ \psi\rangle \xrightarrow{\alpha} \langle B',\ \psi'\rangle \quad \text{et} \quad \mathrm{fv}(\phi)\cap\mathrm{fv}(\psi)=\emptyset}{\langle A|B,\ \phi\wedge\psi\rangle \xrightarrow{\alpha} \langle A|B',\ \phi\wedge\psi'\rangle}$$

Protocole

$$\frac{\langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle, \quad \alpha\notin C \quad \text{et} \quad \mathrm{fv}(\phi)\cap\mathrm{fv}(\psi)=\emptyset}{\langle P\|Q,\ \phi\wedge\psi\rangle \xrightarrow{\alpha} \langle P'\|Q,\ \phi'\wedge\psi\rangle} \quad \text{et}$$

$$\frac{\langle Q,\ \psi\rangle \xrightarrow{\alpha} \langle Q',\ \psi'\rangle, \quad \alpha\notin C \quad \text{et} \quad \mathrm{fv}(\phi)\cap\mathrm{fv}(\psi)=\emptyset}{\langle P\|Q,\ \phi\wedge\psi\rangle \xrightarrow{\alpha} \langle P\|Q',\ \phi\wedge\psi'\rangle}$$

Synchronisation

$$\frac{\langle P,\ \phi\rangle \xrightarrow{\overline{c_{id}}(t)} \langle P',\ \phi'\rangle, \quad \langle Q,\ \psi\rangle \xrightarrow{c_{id'}(x)} \langle Q',\ \psi'\rangle \quad \text{et} \quad \mathrm{fv}(\phi)\cap\mathrm{fv}(\psi)=\emptyset}{\langle P\|Q,\ \phi\wedge\psi\rangle \xrightarrow{\overline{\delta^c_{id}}(t)} \langle P'\#Q,\ \varphi_1\rangle \xrightarrow{\delta^c_{id'}(t)} \langle P'\|Q',\ \varphi_2\rangle} \quad \text{et}$$

$$\frac{\langle Q,\ \phi\rangle \xrightarrow{\overline{c_{id'}}(t)} \langle Q',\ \phi'\rangle, \quad \langle P,\ \psi\rangle \xrightarrow{c_{id}(x)} \langle P',\ \psi'\rangle \quad \text{et} \quad \mathrm{fv}(\phi)\cap\mathrm{fv}(\psi)=\emptyset}{\langle P\|Q,\ \phi\wedge\psi\rangle \xrightarrow{\overline{\delta^c_{id'}}(t)} \langle P\#Q',\ \varphi_1\rangle \xrightarrow{\delta^c_{id}(t)} \langle P'\|Q',\ \varphi_2\rangle}$$

avec $\varphi_1 ::= \phi'\wedge\psi \quad$ et $\quad \varphi_2 ::= \phi'\wedge\psi'\wedge x == t.$

Restriction

$$\frac{\langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle}{\langle P\backslash L,\ \phi\rangle \xrightarrow{\alpha} \langle P'\backslash L,\ \phi'\wedge\phi^L_\alpha\rangle}$$

où ϕ^L_α est telle que $\quad \forall_\varrho\ (\varrho \models \phi^L_\alpha \quad \text{ssi} \quad \varrho(\alpha)\in Act\setminus L).$

Observation

$$\frac{\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle \quad \text{et} \quad \alpha\in\mathcal{O}(\gamma)}{\langle P/\mathcal{O},\ \phi\rangle \xrightarrow{\alpha} \langle P'/\mathcal{O},\ \phi'\rangle}$$

FIG. 4.2 – Sémantique des processus contraints.

2. Le processus P' est un *sous-processus* de $\langle P,\ \phi\rangle$ s'il existe une formule ϕ' telle que $\langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle)$.

Définition 4.22. Un calcul $\langle P_0,\ \phi_0\rangle \xrightarrow{\alpha_0} \langle P_1,\ \phi_1\rangle \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} \langle P_n,\ \phi_n\rangle$ est *minimal* si $P_i \neq P_j$ pour $i \neq j$, sauf si $i = 0$ et $j = n$.

Ainsi, un calcul $\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle$ est minimal s'il ne contient pas de boucle, à moins que celle-ci lie les extrémités $\langle P,\ \phi\rangle$ et $\langle P',\ \phi'\rangle$. Tout calcul $\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P',\ \phi'\rangle$ se réduit clairement à un sous-calcul minimal $\langle P,\ \phi\rangle \xrightarrow{\gamma'} \langle P',\ \phi'\rangle$ (où γ' est une sous-suite de γ) en supprimant toute boucle présente dans le calcul de γ.

Exemple 4.23. Considérons les processus SPPA suivants :

$$A ::= c(x_1).A_1, \quad A_1 ::= c(x_2).A_2 \quad \text{et} \quad A_2 ::= [x_1 = x_2]\,\overline{c}(x_1).A_1\ .$$

La sémantique du processus contraint $\langle A,\ \mathbf{1}\rangle$ est illustrée à la Figure 4.3. Notons que la règle de sémantique `Comparaison` engendre la transition

$$\langle A_2,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle \xrightarrow{\overline{c}_{id_A}(x_1)} \langle A_1,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2) \wedge x_1 == x_2\rangle,$$

mais nous écrivons $\langle A_2,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle \xrightarrow{\overline{c}_{id_A}(x_1)} \langle A_1,\ x_1 == x_2\rangle$ puisque

$$\mathcal{M}(x_1) \wedge \mathcal{M}(x_2) \wedge x_1 == x_2 \quad \overset{\mathcal{V}}{\Leftrightarrow} \quad x_1 == x_2,$$

d'où $\langle A_1,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2) \wedge x_1 == x_2\rangle = \langle A_1,\ x_1 == x_2\rangle$. De façon analogue, la transition

$$\langle A_1,\ x_1 == x_2\rangle \xrightarrow{\overline{c}_{id_A}(x_1)} \langle A_2,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$$

provient de la règle de sémantique `Entrée` et du fait que

$$(\exists_{x_2}\ x_1 == x_2) \wedge \mathcal{M}(x_2) \quad \overset{\mathcal{V}}{\Leftrightarrow} \quad \mathcal{M}(x_1) \wedge \mathcal{M}(x_2).$$

$$\langle A,\ \mathbf{1}\rangle \xrightarrow{c_{id_A}(x_1)} \langle A_1,\ \mathcal{M}(x_1)\rangle \xrightarrow{c_{id_A}(x_2)} \langle A_2,\ \mathcal{M}(x_1)\wedge\mathcal{M}(x_2)\rangle$$
$$\Big\downarrow \overline{c}_{id_A}(x_1) \quad \Big\uparrow c_{id_A}(x_2)$$
$$\langle A_1,\ x_1 == x_2\rangle$$

FIG. 4.3 – Sémantique symbolique de $\langle A,\ \mathbf{1}\rangle$.

Exemple 4.24. Considérons les processus SPPA suivants :

$$B ::= c(y_2).B_1, \quad B_1 ::= \texttt{let}\ y_3 = \mathrm{enc}(y_1, y_2)\ \texttt{in}\ B_2 \quad \text{et} \quad B_2 ::= \overline{c}(y_3).\mathbf{0}$$

où $\mathrm{fv}(B) = \{y_1\}$. Pour les besoins de cet exemple, nous supposons que la variable libre y_1 représente un message quelconque. La sémantique symbolique du processus contraint $\langle B,\ \mathcal{M}(y_1)\rangle$ est donnée à la Figure 4.4, où

$$\phi ::= \mathcal{K}(y_1) \wedge \mathcal{M}(y_2) \wedge y_3 == \{y_2\}_{y_1}.$$

$$\langle B,\ \mathcal{M}(y_1)\rangle \xrightarrow{c_{id_B}(y_2)} \langle B_1,\ \mathcal{M}(y_1)\wedge\mathcal{M}(y_2)\rangle \xrightarrow{\mathrm{enc}_{id_B}} \langle B_2,\ \phi\rangle \xrightarrow{\overline{c_{id_B}}(y_3)} \langle \mathbf{0},\ \phi\rangle$$

FIG. 4.4 – Sémantique symbolique de $\langle B,\ \mathcal{M}(y_1)\rangle$.

Exemple 4.25. Pour cet exemple, nous considérons le protocole, de même que sa spécification, présentés à l'exemple 4.20. La sémantique symbolique des processus contraints $\langle A,\ \mathbf{1}\rangle$ et $\langle B,\ \mathbf{1}\rangle$, représentant les participants du protocole, est illustrée à la Figure 4.5, où $\phi ::= \mathcal{N}(x_1) \wedge (x_2 == \{x_1\}_{k_B})$. La sémantique symbolique du processus contraint $\langle P,\ \mathbf{1}\rangle$, représentant le protocole, est illustrée à la Figure 4.6, où $\psi ::= \phi \wedge (y_1 == x_2) \wedge (y_1 == \{y_2\}_{k_B})$.

Si nous considérons la spécification $P' ::= A' \parallel B$ (toujours de l'Exemple 4.20), où A' contient une variable libre x à la place de k_B, alors nous obtenons les processus

$$\langle A,\ \mathbf{1}\rangle \xrightarrow{\text{newNumber}_{id_A}} \langle A_1,\ \mathcal{N}(x_1)\rangle \xrightarrow{\text{enc}_{id_A}} \langle A_2,\ \phi\rangle \xrightarrow{\overline{c_{id_A}}(x_2)} \langle \mathbf{0},\ \phi\rangle$$

$$\langle B,\ \mathbf{1}\rangle \xrightarrow{c_{id_B}(y_1)} \langle B_1,\ \mathcal{M}(y_1)\rangle \xrightarrow{\text{dec}_{id_B}} \langle \mathbf{0},\ y_1 == \{y_2\}_{k_B}\rangle$$

FIG. 4.5 – Sémantique symbolique de $\langle A,\ \mathbf{1}\rangle$ et $\langle B,\ \mathbf{1}\rangle$.

$$\begin{array}{l} \langle A \parallel B,\ \mathbf{1}\rangle \xrightarrow{\text{newNumber}_{id_A}} \langle A_1 \parallel B,\ \mathcal{N}(x_1)\rangle \xrightarrow{\text{enc}_{id_A}} \langle A_2 \parallel B,\ \phi\rangle \xrightarrow{\overline{\delta^c_{id_A}}(x_2)} \langle A_2 \# B,\ \phi\rangle \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Big\downarrow \delta^c_{id_B}(x_2) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \langle \mathbf{0} \parallel \mathbf{0},\ \psi\rangle \xleftarrow{\text{dec}_{id_B}} \langle \mathbf{0} \parallel B_1,\ \phi \wedge (y_1 == x_2)\rangle \end{array}$$

FIG. 4.6 – Sémantique symbolique de $\langle P,\ \mathbf{1}\rangle$.

contraints $\langle A',\ \mathcal{K}(x)\rangle$ et $\langle P',\ \mathcal{K}(x)\rangle$. La sémantique de ces deux processus contraints est illustrée à la Figure 4.7, avec

$$\langle A',\ \mathcal{K}(x)\rangle \xrightarrow{\text{newNumber}_{id_A}} \langle A'_1,\ \mathcal{N}(x_1) \wedge \mathcal{K}(x)\rangle \xrightarrow{\text{enc}_{id_A}} \langle A'_2,\ \phi'\rangle \xrightarrow{\overline{c_{id_A}}(x_2)} \langle \mathbf{0},\ \phi'\rangle$$

$$\begin{array}{l} \langle A' \parallel B,\ \mathcal{K}(x)\rangle \xrightarrow{\text{newNumber}_{id_A}} \langle A'_1 \parallel B,\ \mathcal{K}(x) \wedge \mathcal{N}(x_1)\rangle \xrightarrow{\text{enc}_{id_A}} \langle A'_2 \parallel B,\ \phi'\rangle \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Big\downarrow \overline{\delta^c_{id_A}}(x_2) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \langle \mathbf{0} \parallel \mathbf{0},\ \psi'\rangle \xleftarrow{\text{dec}_{id_B}} \langle \mathbf{0} \parallel B_1,\ \phi \wedge (y_1 == x_2)\rangle \xleftarrow{\delta^c_{id_B}(x_2)} \langle A'_2 \# B,\ \phi'\rangle \end{array}$$

FIG. 4.7 – Sémantique symbolique de $\langle A',\ \mathcal{K}(x)\rangle$ et $\langle P',\ \mathcal{K}(x)\rangle$.

$$\phi' ::= \mathcal{K}(x) \wedge \mathcal{N}(x_1) \wedge (x_2 == \{x_1\}_x)$$

et

$$\psi' ::= \phi' \wedge (y_1 == x_2) \wedge (y_1 == \{y_2\}_{k_B}).$$

Notons que cette dernière formule est à son tour équivalente à la formule $\mathcal{K}(x) \wedge \mathcal{N}(x_1) \wedge (x_2 == \{x_1\}_x) \wedge (y_1 == x_2) \wedge (x_1 == y_2) \wedge (x == k_B)$.

4.5 Finitude de la sémantique symbolique

Dans cette section, nous démontrons que le graphe de transitions associé à un processus contraint est toujours fini.

Lemme 4.26. *Tout processus contraint* $\langle P,\ \phi\rangle$ *possède un nombre fini de sous-processus, c'est-à-dire l'ensemble* $\{P' \mid \langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle)$ *pour un certain* $\phi'\}$ *est fini.*

Démonstration. Ce résultat provient du fait que les règles de la sémantique symbolique de SPPA (voir Figure 4.1 et Figure 4.2) n'altèrent pas la définition initiale de P (ni celles de ses sous-processus). En effet, ces règles ne font jamais usage de substitutions $P[t/x]$. Ainsi, tout sous-processus P' apparaissant dans $\mathcal{D}(\langle P,\ \phi\rangle)$ doit être syntaxiquement identique à la façon dont il fut initialement défini dans P. Nous pouvons donc conclure que la cardinalité de l'ensemble $\{P' \mid \langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle)$ pour un certain $\phi'\}$ est borné par N_u+2N_b, où N_u est le nombre d'opérateurs SPPA unaires (sortie, entrée, appel de fonction, comparaison, restriction, etc.) utilisés dans la définition syntaxique de P, et N_b est le nombre d'opérateurs SPPA binaires (somme, composition parallèle, etc.) utilisés dans la définition syntaxique de P. □

Remarque 4.27. Du Lemme 4.26, nous pouvons déduire que, pour tout processus contraint $\langle P,\ \phi\rangle$, il n'y a qu'un nombre fini de variables apparaissant dans P et ses sous-processus P'. De plus, ces variables sont exactement celles utilisées dans la définition syntaxique de P. Soit $\{x_1, \ldots, x_n\}$ l'ensemble fini contenant ces variables. En observant les règles de la sémantique symbolique de SPPA, nous pouvons également constater que $\mathrm{fv}(\phi') \subseteq \{x_1, \ldots, x_n\}$ pour toute formule $\phi' \in \mathcal{J}$, avec $\mathcal{J} = \{\phi' \mid \langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle)$ pour un certain $P'\}$. Ainsi, toute variable apparaissant dans une dérivée $\langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle)$ (soit dans P' ou dans ϕ') doit aussi apparaître en quelque part dans la définition initiale de P (ou de l'un de ses sous-processus).

Lemme 4.28. *Pour tout processus contraint $\langle P,\ \phi\rangle$, le nombre de calculs minimaux $\langle P,\ \phi\rangle \xrightarrow{\gamma} \langle P,\ \phi'\rangle$ est fini.*

Démonstration. D'abord, nous voyons que tout processus contraint $\langle P,\ \phi\rangle$ possède qu'un nombre fini de transitions sortantes, c'est-à-dire l'ensemble

$$\{\alpha \in Act \mid \langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle \text{ pour un certain } \langle P',\ \phi'\rangle\}$$

est fini. En effet, nous constatons à partir des règles de sémantique que l'existence d'une transition $\langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle$, de même que la valeur de l'action α, dépend exclusivement du processus P, et non de la formule ϕ (à moins que celle-ci soit équivalente à $\mathbf{0}$). De plus, puisque $\langle P,\ \phi\rangle$ possède seulement un nombre fini de sous-processus (par le Lemme 4.26), le nombre total d'actions α apparaissant dans la sémantique de $\langle P,\ \phi\rangle$ doit nécessairement être fini.

Nous pouvons donc conclure qu'étant donnés deux processus contraints quelconques, il existe un nombre fini de calculs minimaux entre eux. En effet, si N_p désigne le nombre de sous-processus de P et N_a désigne le nombre d'actions apparaissant dans le graphe de transitions $\langle P,\ \phi\rangle$, alors le nombre de calculs minimaux entre $\langle P,\ \phi\rangle$ et n'importe quel autre processus contraint est au plus $N_p!{N_p}^{N_a+1}$, où

- $N_p!$ est une borne supérieure sur le nombre de suites possibles de sous-processus correspondant à un calcul minimal entre $\langle P,\ \phi\rangle$ et un autre processus contraint (aucun sous-processus ne peut apparaître deux fois) ;
- ${N_p}^{N_a+1}$ est une borne supérieure sur le nombre de suites possibles d'actions correspondant à un calcul minimal entre $\langle P,\ \phi\rangle$ et un autre processus contraint.

En particulier, il existe seulement un nombre fini de calculs minimaux entre $\langle P,\ \phi\rangle$ et $\langle P,\ \phi'\rangle$. □

Lemme 4.29. *Si $\mathcal{D}(\langle P,\ \phi\rangle)$ est infini, alors il existe un processus contraint $\langle P_1,\ \phi_1\rangle$ et une suite infini de calcul minimaux*

$$\langle P_1,\ \phi_1\rangle \xrightarrow{\gamma_1} \langle P_1,\ \phi_2\rangle \xrightarrow{\gamma_2} \ldots \langle P_1,\ \phi_l\rangle \xrightarrow{\gamma_l} \langle P_1,\ \phi_{l+1}\rangle \xrightarrow{\gamma_{l+1}} \ldots$$

avec $\mathrm{fv}(\phi_l) = \{x_1, \ldots, x_n\}$ *et* $\gamma_l \in \{\gamma'_1, \gamma'_2, \ldots, \gamma'_m\}$ *pour tout* $l \geq 1$, *tel que chaque* $\gamma'_k \in Act^*$ *apparaît une infinité de fois.*

Démonstration. Si $\mathcal{D}(\langle P,\ \phi\rangle)$ est infini, alors il existe un calcul infini

$$\xi_1 \ = \ \langle P,\ \phi\rangle \xrightarrow{\alpha} \langle P',\ \phi'\rangle \xrightarrow{\alpha'} \ldots \tag{4.1}$$

composé de processus contraints distincts deux-à-deux (c'est-à-dire aucun processus contraint n'apparaît à plus d'une reprise dans le calcul).

Soit $\{x_1, \ldots, x_n\}$ l'ensemble des variables apparaissant dans le calcul ξ_1 (Eq. 4.1) (cet ensemble est fini par la Remarque 4.27). Puisque toute formule ψ apparaissant dans le calcul ξ_1 est telle que $\mathrm{fv}(\psi) \subseteq \{x_1, \ldots, x_n\}$, nous pouvons supposer que ce calcul infini possède un suffixe

$$\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha_1} \langle P_2,\ \psi_2\rangle \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_{k-1}} \langle P_k,\ \psi_k\rangle \xrightarrow{\alpha_k} \ldots$$

tel que $\mathrm{fv}(\phi_1) = \mathrm{fv}(\psi_k) = \{x_1, \ldots, x_n\}$, pour $k \geq 2$. De plus, puisque P possède un nombre fini de sous-processus (par le Lemme 4.26), nous pouvons aussi supposer que le processus P_1 apparaît une infinité de fois dans le calcul. D'où, nous pouvons écrire

$$\xi_2 \ = \ \langle P_1,\ \phi_1\rangle \xrightarrow{\gamma_1} \langle P_1,\ \phi_2\rangle \xrightarrow{\gamma_2} \ldots \xrightarrow{\gamma_{l-1}} \langle P_1,\ \phi_l\rangle \xrightarrow{\gamma_l} \ldots \tag{4.2}$$

avec $\gamma_l \in Act^*$ et $\mathrm{fv}(\phi_l) = \{x_1, \ldots, x_n\}$, pour $l \geq 1$. Nous pouvons aussi supposer que le calcul $\langle P_1,\ \phi_l\rangle \xrightarrow{\gamma_l} \langle P_1,\ \phi_{l+1}\rangle$ est minimal pour tout $l \geq 1$.

D'autre part, par le Lemme 4.28, pour tout $l \geq 1$, il n'y a qu'un nombre fini de calculs minimaux entre $\langle P_1,\ \phi_l\rangle$ et $\langle P_1,\ \phi_{l'}\rangle$, d'où il n'y a qu'un nombre fini de suites d'actions différentes γ_l. Supposons que l'ensemble de ces suites d'actions est donné par $\{\gamma'_1, \gamma'_2, \ldots, \gamma'_m\}$, c'est-à-dire toute suite d'actions γ_l provenant du calcul ξ_2 (Eq. 4.2) est tel que $\gamma_l \in \{\gamma'_1, \gamma'_2, \ldots, \gamma'_m\}$. Finalement, nous pouvons supposer que chaque γ'_k apparaît un infinité de fois dans le calcul ξ_2. En effet, si une suite d'actions γ'_k apparaît seulement un nombre fini de fois, alors il suffit de considérer le calcul infini obtenu en coupant le calcul ξ_2 après la dernière occurrence de γ'_k ; le

calcul ainsi obtenu ne contient aucune occurrence de la suite d'actions γ'_k. □

Notation. Afin de simplifier la présentation, nous utiliserons les notations suivantes :

- nous écrivons $\exists_{y_i^{(k)}}$ à la place de $\exists_{y_1^{(k)},\ldots,y_{n_k}^{(k)}}$, et
- nous écrivons $\phi[y_i^{(k)}]$ à la place de $\phi[y_1^{(k)}/x_1^{(k)}]\ldots[y_{n_k}^{(k)}/x_{n_k}^{(k)}]$.

Définition 4.30. Soient deux familles de formules $\{\psi_k\}_{k=1}^m$ et $\{\psi'_k\}_{k=1}^m$. La famille $\{\Gamma_k\}_{k=1}^m$ de transformations de formules est définie comme suit :

$$\Gamma_k : \ \phi \ \mapsto \ \exists_{y_i^{(k)}}(\phi[y_i^{(k)}] \ \wedge \ \psi_k[y_i^{(k)}] \ \wedge \ \psi'_k) \tag{4.3}$$

où les $y_i^{(k)}$ sont toujours de nouvelles variables, c'est-à-dire les transformations Γ_k n'utilisent jamais les mêmes variables $y_1^{(k)},\ldots,y_{n_k}^{(k)}$ à plus d'une reprise.

Lemme 4.31. *Étant donné un calcul infini*

$$\xi \ = \ \langle P_1,\ \phi_1\rangle \xrightarrow{\gamma_1} \langle P_1,\ \phi_2\rangle \xrightarrow{\gamma_2} \ldots \ \langle P_1,\ \phi_l\rangle \xrightarrow{\gamma_l} \langle P_1,\ \phi_{l+1}\rangle \ldots \tag{4.4}$$

satisfaisant les propriétés exposées au Lemme 4.29, il existe des formules ψ_k et ψ'_k (pour $k = 1,\ldots,m$) telles que, pour tout $l \geq 1$,

$$\phi_{l+1} \Leftrightarrow \Gamma_k(\phi_l)$$

dès que $\gamma_l = \gamma'_k$.

Démonstration. Du Lemme 4.28, nous savons que la capacité d'un processus contraint $\langle P',\ \phi'\rangle$ d'exécuter une action dépend uniquement de la définition du processus P' (aussi longtemps que ϕ' n'est pas équivalente à $\mathbf{0}$). D'où, étant donnée $\gamma'_k \in Act^*$, tout calcul $\langle P_1,\ \phi_l\rangle \xrightarrow{\gamma'_k} \langle P_1,\ \phi_{l+1}\rangle$ transforme la formule ϕ_l en la formule ϕ_{l+1} selon une règle unique, peu importe l. De plus, des règles de la sémantique symbolique de SPPA, nous pouvons constater que

$$\phi_{l+1} ::= \ \exists_{x_1^{(k)},\ldots,x_{n_k}^{(k)}}(\phi_l \wedge \psi_k) \ \wedge \ \psi'_k$$

pour certaines formules ψ_k et ψ'_k (qui ne dépendent pas de l), avec $\{x_1^{(k)}, \ldots, x_{n_k}^{(k)}\} \subseteq \{x_1, \ldots, x_n\}$. Il suffit maintenant de noter que la formule ϕ_{l+1} est équivalente à la formule

$$\exists_{y_1^{(k)},\ldots,y_{n_k}^{(k)}}(\phi_l[y_1^{(k)}/x_1^{(k)}]\ldots[y_{n_k}^{(k)}/x_{n_k}^{(k)}] \ \wedge \ \psi_k[y_1^{(k)}/x_1^{(k)}]\ldots[y_{n_k}^{(k)}/x_{n_k}^{(k)}] \ \wedge \ \psi'_k),$$

où les $y_i^{(k)}$ sont de nouvelles variables. Si nous réécrivons cette formule à l'aide de la notation établie plus au haut, nous obtenons

$$\phi_{l+1} \Leftrightarrow \exists_{y_i^{(k)}}(\phi_l[y_i^{(k)}] \ \wedge \ \psi_k[y_i^{(k)}] \ \wedge \ \psi'_k).$$

Si nous exprimons cette formule en terme de le famille de transformations $\{\Gamma_k\}_{k=1}^m$, nous avons $\Gamma_k(\phi_l) \Leftrightarrow \phi_{l+1}$ pour tout $l \geq 1$ tel que $\langle P_1, \ \phi_l\rangle \xrightarrow{\gamma'_k} \langle P_1, \ \phi_{l+1}\rangle$. □

Pour le lemme suivant, nous considérons des formules ψ_k et ψ'_k, ainsi que des transformations Γ_k, définies à partir de celles-ci, pour $k = 1, \ldots, m$.

Lemme 4.32. *Pour toute suite infinie de formules $\varsigma = \{\phi_l\}_{l\geq 1}$ telles que*

$$\phi_l = \Gamma_{k_l}(\phi_{l-1}) \quad \textit{avec } 1 \leq k_l \leq m$$

et où chaque transformation Γ_k (pour $k = 1, \ldots, m$) revient une infinité de fois, il existe un entier $K \geq 1$ et une famille de formules $\{\theta_k\}_{k=1}^m$ telles que, pour tout $l \geq K$, $\Gamma_k(\phi_l) \Leftrightarrow \theta_k$ dès que $k_{l+1} = k$.

Démonstration. Toute transformation Γ_k fait essentiellement deux actions : d'une part, Γ_k substitue les variables $x_1^{(k)}, \ldots, x_{n_k}^{(k)}$ avec de nouvelles variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$, et, d'autre part, Γ_k introduit (à l'aide d'une conjonction) les formules ψ_k et ψ'_k. Mais, puisque Γ_k introduit toujours de nouvelles variables, nous pouvons supposer que toute formule ϕ_l de la suite ς est de la forme $\exists_{y_1,\ldots,y_n}\psi$, où les $y_1, \ldots, y_n$ sont des variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$ introduites lors d'applications précédentes de transformations Γ_k (pour $1 \leq k \leq m$), et ψ est une certaine formule sans quantificateur. De plus, nous pouvons constater que la formule ψ doit être de la forme $\phi_1[y_i^{(k_1)}]\ldots[y_i^{(k_l)}] \wedge \psi'$,

où la formule ψ' est une conjonction de formules ψ_k et ψ'_k (pour $1 \leq k \leq m$) sur lesquelles des substitutions $[y_i^{(k_1)}] \ldots [y_i^{(k_l)}]$ furent appliquées.

Malgré le fait qu'il y aura une infinité de nouvelles variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$ introduites lors du calcul infini, nous démontrons qu'à partir d'un certain moment dans le calcul, l'introduction de ces nouvelles variables par un Γ_k, et leur substitution avec les variables $x_1^{(k)}, \ldots, x_{n_k}^{(k)}$, créera inévitablement une formule ϕ_l équivalente à une formule précédente $\phi_{l'}$.

D'abord, nous voyons que les substitutions introduites par Γ_k sont appliquées au plus une fois sur les formules dans ψ'. En effet, des substitutions consécutives

$$\phi_l[y_1^{(k)}/x_1^{(k)}] \ldots [y_{n_k}^{(k)}/x_{n_k}^{(k)}][z_1^{(k)}/x_1^{(k)}] \ldots [z_{n_k}^{(k)}/x_{n_k}^{(k)}]$$

donnent la formule $\phi_l[y_1^{(k)}/x_1^{(k)}] \ldots [y_{n_k}^{(k)}/x_{n_k}^{(k)}]$, c'est-à-dire $\phi_l[y_i^{(k)}]$. La même remarque est également vraie pour toutes formules ψ_k et ψ'_k, pour $1 \leq k \leq m$, et conséquemment pour la formule ψ'. D'où, puisqu'il y a m transformations Γ_k, donc m ensembles de variables $\{x_1^{(k)}, \ldots, x_{n_k}^{(k)}\}$ soumises aux substitutions, il y a au plus $m!$ substitutions composées non équivalentes de la forme $[y_i^{(k_1)}] \ldots [y_i^{(k_l)}]$ qui peuvent être obtenues de la composition des Γ_k.[10] De plus, puisque chaque transformation Γ_k introduit toujours les mêmes formules ψ_k et ψ'_k, le nombre de formules possibles que nous pouvons obtenir des ψ_k, ψ'_k et des substitutions composées sur les ensembles de variables $\{x_1^{(k)}, \ldots, x_{n_k}^{(k)}\}$ est borné par $2m(m!)$, donc fini.

D'autre part, puisque que chaque transformation Γ_k revient infiniment souvent dans la suite ς, à un certain point, pour toutes nouvelles variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$ introduites par une certaine transformation Γ_k (à travers une substitution des variables $x_1^{(k)}, \ldots, x_{n_k}^{(k)}$), il existe des variables $z_1^{(k)}, \ldots, z_{n_k}^{(k)}$ introduites préalablement qui sont présentes exactement dans les mêmes substitutions composées équivalentes appliquées exactement aux mêmes formules ψ_l et ψ'_l. Dans ce cas, ces nouvelles variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$ peuvent être remplacées par les $z_1^{(k)}, \ldots, z_{n_k}^{(k)}$, et la formule ainsi obtenue

[10] deux substitutions composées sont équivalentes lorsque l'ordre selon laquelle les ensembles de variables $\{x_1^{(k)}, \ldots, x_{n_k}^{(k)}\}$ sont substituées est exactement le même (modulo les noms des nouvelles variables $y_1^{(k)}, \ldots, y_{n_k}^{(k)}$).

est équivalente à la formule précédente ϕ_l.

En fait, nous pouvons constater qu'à partir d'un certain point dans la suite ς, plus précisément lorsque chaque transformation Γ_k est appliquée au moins $2m(m_1)!$ fois, toute nouvelle application nous donne une formule équivalente à la formule obtenue de l'application précédente de Γ_k. Ainsi, il existe des formules $\theta_1, \ldots, \theta_m$ telles que $\phi_{l+1} \Leftrightarrow \theta_k$ lorsque

$$\langle P_1,\ \phi_l \rangle \xrightarrow{\gamma'_k} \langle P_1,\ \phi_{l+1} \rangle$$

pour tout $l \geq K$, pour un certain $K \geq 0$ suffisamment grand. D'où $\Gamma_k(\theta_{k'}) \Leftrightarrow \theta_k$.

□

Remarque 4.33. Les formules θ_k mentionnées dans la preuve précédente sont trop longues pour être écrites explicitement dans ce document, mais nous pouvons constater qu'elles ont la forme suivante :

$$\theta_k ::= \exists_{y_1,\ldots,y_{n'}} (\ \phi_1[y_i^{(k_1)}] \ldots [y_i^{(k_m)}]\ \wedge\ \theta\ \wedge\ \psi_k[z_i^{(k)}]\ \wedge\ \psi'_k\ )$$

où $y_1, \ldots, y_{n'}$ sont de nouvelles variables introduites par la transformation Γ_k, avec $\{k_1, \ldots, k_m\} = \{1, \ldots, m\}$ et $y_i^{(k_1)}, \ldots, y_i^{(k_m)}, z_i^{(k)} \in \{y_1, \ldots, y_{n'}\}$. La formule θ, présente dans toutes les θ_k, est la formule obtenue en considérant la conjonction de toutes les combinaisons possibles de substitutions composées $[y_i^{(k_1)}] \ldots [y_i^{(k_m)}]$ (pour de nouvelles variables $y_i^{(k)}$) appliquées sur toutes formules ψ_k et ψ'_k (pour $k = 1, \ldots, m$). La formule θ s'avère être un point fixe pour toute transformation Γ_k. En effet, puisque θ contient toutes les substitutions composées appliquées à toutes les formules introduites par les transformations, alors toute nouvelle substitution introduite par un certain Γ_k n'aura aucun effet sur θ. La famille de formules $\{\theta_k\}_{k=1}^m$ est donc telle que $\Gamma_k(\theta_{k'}) \Leftrightarrow \theta_k$, pour tout k, k'.

Théorème 4.34. *Pour tout processus contraint $\langle P,\ \phi \rangle$, l'ensemble $\mathcal{D}(\langle P,\ \phi \rangle)$ est fini.*

Démonstration. Supposons que $\langle P,\ \phi \rangle$ est un processus contraint ayant une infi-

nité de dérivées, c'est-à-dire $\mathcal{D}(\langle P,\ \phi\rangle)$ est infini. Par le Lemme 4.29, il existe un processus contraint $\langle P_1,\ \phi_1\rangle$ et un calcul infini

$$\xi\ =\ \langle P_1,\ \phi_1\rangle \xrightarrow{\gamma_1} \langle P_1,\ \phi_2\rangle \xrightarrow{\gamma_2} \ldots\ \langle P_1,\ \phi_l\rangle \xrightarrow{\gamma_l} \langle P_1,\ \phi_{l+1}\rangle \xrightarrow{\gamma_{l+1}} \ldots \qquad (4.5)$$

avec $\mathrm{fv}(\phi_l) = \{x_1, \ldots, x_n\}$ et $\gamma_l \in \{\gamma'_1, \gamma'_2, \ldots, \gamma'_m\}$ pour tout $l \geq 1$, tel que chaque $\gamma'_k \in Act^*$ est minimal et apparaît une infinité de fois. De plus, par le Lemme 4.31, il existe des familles de formules $\{\psi_k\}_{k=1}^m$ et $\{\psi'_k\}_{k=1}^m$ telles que

$$\phi_{l+1} \Leftrightarrow \Gamma_k(\phi_l)$$

dès que $\gamma_l = \gamma'_k$. Ainsi, la suite de formules $\phi_1, \phi_2, \phi_3, \ldots$ provenant du calcul ξ_2 (Eq. 4.2) peut s'écrire de la façon suivante :

$$\varsigma = \{\phi_1,\ \Gamma_{k_1}(\phi_1),\ \Gamma_{k_2}(\Gamma_{k_1}(\phi_1)),\ \Gamma_{k_3}(\Gamma_{k_2}(\Gamma_{k_1}(\phi_1))), \ldots\}$$

avec $\gamma_l = \gamma'_{k_l}$, et où chaque transformation Γ_k revient une infinité de fois (car chaque γ'_k apparaît une infinité de fois dans le calcul ξ (Eq. 4.5)).

Par le Lemme 4.32, il existe un entier $K \geq 1$ et des formules θ_k (pour $k = 1, \ldots, m$) telles que, pour tout $l \geq K$, $\Gamma_k(\phi_l) \Leftrightarrow \theta_k$ dès que $k_{l+1} = k$. Mais ceci constitue une contradiction au fait que les formules ϕ_l sont deux-à-deux non équivalentes, donc une contradiction à l'existence du calcul infini ξ. Par conséquent, nous devons conclure que l'ensemble $\mathcal{D}(\langle P,\ \phi\rangle)$ est fini. □

4.5.1 Sémantique symbolique vs sémantique avec passage de paramètres

Les relations reliant la sémantique symbolique opérationnelle d'un processus contraint et la sémantique opérationnelle avec passage de paramètres d'un processus SPPA sont exposées en détail dans les lemmes suivants. En bref, toute suite de transitions entre deux processus SPPA correspond à une suite de transitions entre deux processus contraints. Réciproquement, toute transition entre deux processus contraints correspond à un ensemble de transitions entre deux processus SPPA.

La prochaine définition nous permet d'établir une distinction entre deux types d'actions.

Définition 4.35. Les actions SPPA de la forme $\overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} et τ sont dites de *Type I*, et les actions SPPA de la forme $c_{id}(x)$, f_{id}, split_{id} et dec_{id} sont dites de *Type II*. Les actions de Type I sont désignées par α, et celles de Type II sont désignées par β.

Du point de vue de la sémantique symbolique, nous voyons qu'une action de Type II est caractérisée par l'introduction d'une nouvelle variable x (ou deux, x et y, dans le cas de l'action split_{id}), ce qui n'est pas le cas pour une action de Type I.

Hypothèse. Pour le reste du chapitre, nous supposons que, étant donnée une action β de Type II, x est toujours la variable introduite par cette action. De plus, par souci de simplicité, nous traitons le cas de l'action split_{id} qui introduit deux variables au même titre que les autres actions β. Nous pouvons facilement constater que cette dernière hypothèse n'affectera pas les résultats présentés dans ce chapitre car des preuves complètes s'obtiennent simplement en ajoutant, au besoin, des cas spéciaux pour l'action split_{id}.

Le lemme suivant nous assure que toute action dans la sémantique d'un processus SPPA correspond à une action de la sémantique symbolique d'un processus contraint.

Lemme 4.36. *Soient P, P' des processus SPPA, soit $\alpha' = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} ou τ, pour un certain terme t tel que $\text{fv}(t) \subseteq \{x_1, \ldots, x_n\}$, et soit $\beta' = c_{id}(x)$, f_{id}, split_{id} ou dec_{id}.*

- *Si $P[a_1/x_1]\ldots[a_n/x_n] \xrightarrow{\alpha} P'[a_1/x_1]\ldots[a_n/x_n]$, pour certains $a_1, \ldots, a_n \in \mathcal{M}$ $(n \geq 0)$ et $\alpha = \alpha'[a_1/x_1]\ldots[a_n/x_n]$, alors, pour toute formule $\phi \not\Leftrightarrow \mathbf{0}$ telle que $\text{fv}(\phi) = \{x_1, \ldots, x_n\}$, nous avons*

$$\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle$$

où ϕ' est la formule donnée par les règles de la sémantique symbolique (avec $\phi' \not\approx \mathbf{0}$).

- *Si* $P[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\beta} P'[a_1/x_1]\dots[a_n/x_n][a/x]$, *pour certains* $a_1, \dots, a_n, a \in \mathcal{M}$ $(n \geq 0)$ *et* $\beta = \beta'[a/x]$, *alors, pour toute formule* $\phi \not\approx \mathbf{0}$ *telle que* $\mathrm{fv}(\phi) = \{x_1, \dots, x_n\}$, *nous avons*

$$\langle P,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle$$

où ϕ' est la formule donnée par les règles de la sémantique symbolique (avec $\phi' \not\approx \mathbf{0}$).

Démonstration. Afin de raccourcir la démonstration, les deux énoncés sont démontrés simultanément par induction sur la structure syntaxique de P. Selon l'énoncé que nous souhaitons démontrer, nous posons

$$Q' ::= P'[a_1/x_1]\dots[a_n/x_n] \quad \text{ou} \quad Q' ::= P'[a_1/x_1]\dots[a_n/x_n][a/x].$$

Tout d'abord, nous constatons que le cas où $P ::= \mathbf{0}$ est trivial. D'autre part, si $P ::= \overline{c}(t).P'$, $c(x).P'$, $\mathtt{let}\ x = f(t)\ \mathtt{in}\ P'$, $\mathtt{let}\ (x,y) = t\ \mathtt{in}\ P'$, $\mathtt{case}\ t\ \mathtt{of}\ \{x\}_{t'}\ \mathtt{in}\ P'$ ou $\mathtt{case}\ t\ \mathtt{of}\ [t'']_{t'}\ \mathtt{in}\ P'$, alors la conclusion suit directement des règles de sémantique `Sortie`, `Entrée`, `Fonction`, `Générateur`, `Extraction`, `Décryption` et `Signature-Vérif`.

Si $P ::= P_1 + P_2$, $P_1|P_2$, ou $P_1 \parallel P_2$ (et α n'est pas une action de marquage), alors, des règles de sémantique `Somme`, `Parallèle` et `Protocole`, nous pouvons supposer que

$$P_1[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\alpha} P_1'[a_1/x_1]\dots[a_n/x_n]$$

$$(\text{resp. } P_1[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\beta} P_1'[a_1/x_1]\dots[a_n/x_n][a/x])$$

avec $P' ::= P_1'$, $P_1'|P_2$, ou $P_1' \parallel P_2$. Donc, par l'hypothèse d'induction,

$$\langle P_1,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle P_1',\ \phi'\rangle.$$

Ainsi, $\langle P,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle P',\ \phi'\rangle$. Si $P ::= P_1 \parallel P_2$ ou $P_1 \# P_2$, et α est une action de marquage, alors nous pouvons supposer que

$$P_1[a_1/x_1]\ldots[a_n/x_n] \xrightarrow{\overline{c_{id_1}}(a)} P_1'[a_1/x_1]\ldots[a_n/x_n]$$

et

$$P_2[a_1/x_1]\ldots[a_n/x_n] \xrightarrow{c_{id_2}(a)} P_2'[a_1/x_1]\ldots[a_n/x_n][a/x]$$

avec $P' ::= P_1 \# P_2$ ou $P' ::= P_1' \parallel P_2'$. Par l'hypothèse d'induction, nous avons $\langle P_1,\ \phi\rangle \xrightarrow{\overline{c_{id_1}}(t)} \langle P_1',\ \phi'\rangle$ (où $a = t[a_1/x_1]\ldots[a_n/x_n]$) et $\langle P_2,\ \phi\rangle \xrightarrow{c_{id_2}(x)} \langle P_2',\ \phi'\rangle$. D'où $\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle$ par la règle `Synchronisation`.

Si $P ::= [t = t']P_1$ (avec $t[a_1/x_1]\ldots[a_n/x_n] = t'[a_1/x_1]\ldots[a_n/x_n]$), alors $P_1[a_1/x_1]\ldots[a_n/x_n] \xrightarrow{\alpha/\beta} Q'$ et, par l'hypothèse d'induction, nous voyons que

$$\langle P_1,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle P',\ \phi'\rangle.$$

Donc, par la règle de sémantique `Comparaison`, $\langle P,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle P',\ \phi'\rangle$ avec $\phi' \not\Leftrightarrow \mathbf{0}$ car $\varrho \models t == t'$ pour toute évaluation ϱ telle que $\varrho(x_i) = a_i$.

Si $P ::= P_1 \setminus L$ (avec $\alpha, \beta \notin L$), alors $P_1[a_1/x_1]\ldots[a_n/x_n] \xrightarrow{\alpha/\beta} Q'$ et, par l'hypothèse d'induction, nous avons $\langle P_1,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle P',\ \phi''\rangle$ avec $\phi'' \not\Leftrightarrow \mathbf{0}$. Ainsi, nous avons $\langle P,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle P',\ \phi'\rangle$ avec $\phi' \not\Leftrightarrow \mathbf{0}$ car $\varrho \models \phi^L_{\alpha'/\beta'}$ pour toute évaluation ϱ telle que $\varrho(x_i) = a_i$.

Finalement, si $P ::= P_1/\mathcal{O}$, alors il existe un calcul

$$P_1[a_1/x_1]\ldots[a_n/x_n] \xrightarrow{\gamma} P_1'[a_1/x_1]\ldots[a_n/x_n]$$

$$(\text{resp. } P_1[a_1/x_1]\ldots[a_n/x_n] \xrightarrow{\gamma} P_1'[a_1/x_1]\ldots[a_n/x_n][a/x])$$

tel que $\alpha \in \mathcal{O}(\gamma)$ (resp. $\beta \in \mathcal{O}(\gamma)$) et avec $Q' ::= P_1'[a_1/x_1]\ldots[a_n/x_n]/\mathcal{O}$ (resp. $Q' ::= P_1'[a_1/x_1]\ldots[a_n/x_n][a/x]/\mathcal{O}$). Donc, par l'hypothèse d'induction, nous avons $\langle P_1,\ \phi\rangle \xrightarrow{\gamma'} \langle P_1',\ \phi'\rangle$ où $\gamma = \gamma'[a_1/x_1]\ldots[a_n/x_n]$ (resp. $\gamma = \gamma'[a_1/x_1]\ldots[a_n/x_n][a/x]$). D'où, par la règle de sémantique `Observation` et

puisque $P ::= P_1/\mathcal{O}$ et $P' ::= P_1'/\mathcal{O}$, nous pouvons constater que $\langle P,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle P',\ \phi'\rangle$, ce qui conclut la démonstration. □

Le lemme suivant nous assure que toute action dans la sémantique symbolique d'un processus contraint correspond à une action de la sémantique d'un processus SPPA.

Lemme 4.37. *Soient P, P' des processus SPPA, soit $\alpha' = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} ou τ, pour un certain terme t tel que $\mathrm{fv}(t) \subseteq \{x_1, \ldots, x_n\}$, et soit $\beta' = c_{id}(x)$, f_{id}, split_{id} ou dec_{id}.*

- *Si $\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle$, avec $\mathrm{fv}(\phi) = \mathrm{fv}(\phi') = \{x_1, \ldots, x_n\}$, alors, pour toute évaluation ϱ telle que $\varrho \models \phi'$,*

$$P[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n] \xrightarrow{\alpha} P'[\varrho(x_1)/x_1] \ldots [\varrho(x_1)/x_n]$$

  où $\alpha = \varrho(\alpha')$.
- *Si $\langle P,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle$, avec $\mathrm{fv}(\phi) = \{x_1, \ldots, x_n\}$ et $\mathrm{fv}(\phi') = \mathrm{fv}(\phi) \cup \{x\}$, alors, pour toute évaluation ϱ telle que $\varrho \models \phi'$,*

$$P[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n] \xrightarrow{\beta} P'[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n][\varrho(x)/x]$$

  où $\beta = \varrho(\beta')$.

Démonstration. Afin de raccourcir la démonstration, les deux énoncés sont démontrés simultanément par induction sur la structure syntaxique de P. Le cas où $P ::= \mathbf{0}$ est trivial. Soit ϱ une évaluation telle que $\varrho \models \phi'$, et posons $\alpha ::= \varrho(\alpha')$, $\beta ::= \varrho(\beta')$ et $Q ::= P[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$. De plus, selon l'énoncé que nous souhaitons démontrer, nous posons

$$Q' ::= P'[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$$

ou

$$Q' ::= P'[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n][\varrho(x)/x].$$

Si $P ::= \overline{c}(t).P'$, $c(x).P'$, $\texttt{let } x = f(t) \texttt{ in } P'$, $\texttt{let } (x,y) = t \texttt{ in } P'$, $\texttt{case } t \texttt{ of } \{x\}_{t'} \texttt{ in } P'$, ou $\texttt{case } t \texttt{ of } [t'']_{t'} \texttt{ in } P'$, alors nous avons, respectivement, $Q ::= \overline{c}(\varrho(t)).Q'$, $c(x).Q'$, $\texttt{let } x = f(\varrho(t)) \texttt{ in } Q'$, $\texttt{let } (x,y) = \varrho(t) \texttt{ in } Q'$, $\texttt{case } \{\varrho(t')\}_{\varrho(t)} \texttt{ of } \{x\}_{\varrho(t)} \texttt{ in } Q'$, ou $\texttt{case } [\varrho(t')]_{\varrho(t)} \texttt{ of } [\varrho(t'')]_{\varrho(t)} \texttt{ in } Q'$. D'où $Q \xrightarrow{\alpha/\beta} Q'$.

Si $P ::= P_1 + P_2$, $P_1|P_2$, $P_1 \parallel P_2$, ou $P_1 \# P_2$, alors nous avons, respectivement, $Q ::= Q_1 + Q_2$, $Q_1|Q_2$, $Q_1 \parallel Q_2$, ou $Q_1 \# Q_2$ avec $Q_1 ::= P_1[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$ et $Q_2 ::= P_2[\varrho(x_1)/x_1] \ldots [\varrho(x_1)/x_n]$. Lorsque α' n'est pas une action de marquage, nous pouvons supposer que $\langle P_1,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle P_1',\ \phi'\rangle$ avec, respectivement, $P' ::= P_1'$, $P_1'|P_2$, ou $P_1' \parallel P_2$. D'où, par l'hypothèse d'induction, nous voyons que

$$Q_1 \xrightarrow{\alpha} P_1'[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$$

$$(\text{resp. } Q_1 \xrightarrow{\beta} P_1'[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n][\varrho(x)/x]),$$

donc $Q \xrightarrow{\alpha/\beta} Q'$. Supposons maintenant que α' est une action de marquage, avec $P ::= P_1 \parallel P_2$ ou $P ::= P_1 \# P_2$. Dans ce cas, nous pouvons supposer que $\langle P_1,\ \psi_1\rangle \xrightarrow{\overline{c_{id_1}}(t)} \langle P_1',\ \psi_1'\rangle$ et $\langle P_2,\ \psi_2\rangle \xrightarrow{c_{id_2}(x)} \langle P_2',\ \psi_2'\rangle$ avec $\phi ::= \psi_1 \wedge \psi_2$. Par l'hypothèse d'induction, nous avons

$$Q_1 \xrightarrow{\overline{c_{id_1}}(\varrho(t))} P_1'[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$$

et

$$Q_2 \xrightarrow{c_{id_2}(\varrho(t))} P_2'[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n][a/x].$$

Donc, $P \xrightarrow{\alpha} P'$ par la règle `Synchronisation`, avec $P ::= P_1 \# P_2$ ou $P ::= P_1' \parallel P_2'$.

Si $P ::= [t = t']P_1$, alors nous devons avoir $\langle P_1,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle P',\ \psi\rangle$ avec $\varrho \models \psi$ et $\varrho \models (t == t')$ car $\varrho \models \phi'$. Ainsi, par l'hypothèse d'induction, nous avons $\langle Q_1,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle Q',\ \psi\rangle$ où $Q_1 ::= P_1[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$. D'où, par la règle de sémantique `Comparaison`, $Q \xrightarrow{\alpha/\beta} Q'$ car $\varrho(t) = \varrho(t')$.

Si $P ::= P_1 \setminus L$, alors nous avons $\langle P_1,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle P_1',\ \psi\rangle$ avec $P' ::= P_1' \setminus L$ et $\phi ::= \psi \wedge \phi^L_{\alpha'}$ (resp. $\phi ::= \psi \wedge \phi^L_{\beta'}$). De plus, nous pouvons constater que $Q ::= Q_1 \setminus L$ où $Q_1 ::= P_1[\varrho(x_1)/x_1] \ldots [\varrho(x_n)/x_n]$. Mais puisque $\varrho \models \psi$, par

l'hypothèse d'induction, nous avons que

$$Q_1 \xrightarrow{\alpha} P_1'[\varrho(x_1)/x_1]\dots[\varrho(x_n)/x_n]$$

$$(\text{resp. } Q_1 \xrightarrow{\beta} P_1'[\varrho(x_1)/x_1]\dots[\varrho(x_n)/x_n][\varrho(x)/x]).$$

D'autre part, puisque $\varrho \models \phi'$, donc $\varrho \models \phi'^{L}_{\alpha'}$ (resp. $\varrho \models \phi'^{L}_{\beta'}$), nous avons que $\alpha, \beta \notin L$. D'où $Q \xrightarrow{\alpha/\beta} Q'$.

Si $P ::= P_1/\mathcal{O}$, alors il existe un calcul $\langle P_1,\ \phi\rangle \xrightarrow{\gamma'} \langle P_1',\ \phi'\rangle$ tel que $\alpha \in \mathcal{O}(\gamma')$ (resp. $\beta \in \mathcal{O}(\gamma')$) avec $P' ::= P_1'/\mathcal{O}$. Par l'hypothèse d'induction, nous voyons que $Q_1 \xrightarrow{\gamma} Q_1'$, où $Q_1' ::= P_1'[\varrho(x_1)/x_1]\dots[\varrho(x_n)/x_n]$ (resp. $Q_1' ::= P_1'[\varrho(x_1)/x_1]\dots[\varrho(x_n)/x_n][\varrho(x)/x]$) et $\gamma = \varrho(\gamma')$. D'où, $Q \xrightarrow{\alpha/\beta} Q'$ car $\alpha \in \mathcal{O}(\gamma)$ (resp. $\beta \in \mathcal{O}(\gamma)$). □

Le prochain lemme nous permet de conclure qu'étant donnés un processus qui peut exécuter une action et un processus contraint dont la formule est satisfaite et qui peut exécuter cette même action, alors la formule dans le processus contraint suivant l'action est nécessairement satisfaite.

Lemme 4.38. *Soient P, P' des processus SPPA, soit $\alpha' = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} ou τ, pour un certain terme t tel que $\mathrm{fv}(t) \subseteq \{x_1, \dots, x_n\}$, et soit $\beta' = c_{id}(x)$, f_{id}, split_{id} ou dec_{id}. Soient $a_1, \dots, a_n, a \in \mathcal{M}$, et posons $\alpha = \alpha'[a_1/x_1]\dots[a_n/x_n]$ et $\beta = \beta'[a/x]$.*

- *Si $P[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\alpha} P'[a_1/x_1]\dots[a_n/x_n]$ et $\models \phi[a_1/x_1]\dots[a_n/x_n]$, alors $\models \phi'[a_1/x_1]\dots[a_n/x_n]$ lorsque*

$$\langle P,\ \phi\rangle \xrightarrow{\alpha'} \langle P',\ \phi'\rangle.$$

- *Si $P[a_1/x_1]\dots[a_n/x_n] \xrightarrow{\beta} P'[a_1/x_1]\dots[a_n/x_n][a/x]$ et $\models \phi[a_1/x_1]\dots[a_n/x_n]$, alors $\models \phi'[a_1/x_1]\dots[a_n/x_n][a/x]$ lorsque*

$$\langle P,\ \phi\rangle \xrightarrow{\beta'} \langle P',\ \phi'\rangle.$$

Démonstration. Tout comme les Lemmes précédents, nous démontrons les deux énoncés simultanément, et pour chacun nous procédons par induction sur la structure syntaxique de P. Si la transition $\langle P,\ \phi\rangle \xrightarrow{\alpha'/\beta'} \langle P',\ \phi'\rangle$ provient des règles `Entrée`, `Sortie`, `Fonction` ou `Générateur`, alors nous pouvons constater directement à partir de la définition de ϕ' que $\models \phi'[a_1/x_1]\dots[a_n/x_n]$ (resp. $\models \phi'[a_1/x_1]\dots[a_n/x_n][a/x]$) lorsque $\models \phi[a_1/x_1]\dots[a_n/x_n]$.

Si $P ::= P_1 + P_2$, $P_1|P_2$ ou $P_1 \parallel P_2$, et α n'es pas une action de marquage, alors nous pouvons supposer que $\phi ::= \phi_1 \wedge \phi_2$ et $\phi' ::= \phi'_1 \wedge \phi_2$, avec $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha'/\beta'} \langle P'_1,\ \phi'_1\rangle$. Ainsi, puisque que $\models \phi[a_1/x_1]\dots[a_n/x_n]$, nous avons $\models \phi_1[a_1/x_1]\dots[a_n/x_n]$, d'où $\models \phi'_1[a_1/x_1]\dots[a_n/x_n]$ (resp. $\models \phi'_1[a_1/x_1]\dots[a_n/x_n][a/x]$) par l'hypothèse d'induction. D'où, $\models \phi'[a_1/x_1]\dots[a_n/x_n][a/x]$ (resp. $\models \phi'[a_1/x_1]\dots[a_n/x_n][a/x]$) car $\models \phi_2[a_1/x_1]\dots[a_n/x_n]$. Les cas où $P ::= P_1 \parallel P_2$ ou $P ::= P_1 \# P_2$, et α est une action de marquage, sont similaires. Si $P ::= [t = t']P_1$, alors nous voyons que l'énoncé demeure valide pour chacune des deux formes possibles pour ϕ', car $t[a_1/x_1]\dots[a_n/x_n] = t'[a_1/x_1]\dots[a_n/x_n]$.

Si $P ::= P_1 \setminus L$, alors $\phi' ::= \phi_1 \wedge \phi^L_{\alpha'}$. Par l'hypothèse d'induction, nous pouvons constater que $\models \phi_1[a_1/x_1]\dots[a_n/x_n]$ (resp. $\models \phi_1[a_1/x_1]\dots[a_n/x_n][a/x]$), et puisque $\alpha \notin L$, $\models \phi^L_{\alpha'}[a_1/x_1]\dots[a_n/x_n]$ (resp. $\models \phi^L_{\beta'}[a_1/x_1]\dots[a_n/x_n][a/x]$). Ainsi, $\models \phi'[a_1/x_1]\dots[a_n/x_n][a/x]$ (resp. $\models \phi'[a_1/x_1]\dots[a_n/x_n][a/x]$). Finalement, supposons que $P ::= P_1/\mathcal{O}$. Alors, $\langle P_1,\ \phi\rangle \xrightarrow{\gamma'} \langle P'_1,\ \phi'\rangle$, où $P' ::= P'_1$ et $\alpha' \in \mathcal{O}(\gamma')$ (resp. $\beta' \in \mathcal{O}(\gamma')$). D'où, en utilisant l'hypothèse d'induction pour chaque sous-action de γ', nous voyons que $\models \phi'[a_1/x_1]\dots[a_n/x_n]$ (resp. $\models \phi'[a_1/x_1]\dots[a_n/x_n][a/x]$) lorsque $\models \phi[a_1/x_1]\dots[a_n/x_n]$. □

En résumé, les Lemmes 4.36 et 4.37 nous permettent d'établir une relation biunivoque entre les actions d'un processus SPPA et les actions d'un processus contraint. D'autre part, nous pouvons grossièrement déduire du Lemme 4.38 la cohérence de la sémantique symbolique des processus contraints : si un processus peut exécuter une action et cette action satisfait une certaine formule, alors le processus contraint composé du processus et de la formule peut exécuter la même

action et évoluer dans un processus contraint dont la formule est encore satisfaite.

4.6 Bisimulation entre processus contraints

Dans cette section, nous étendons la notion de bisimulation [81] (voir Définition 3.18) de Milner afin qu'elle puisse traiter les processus contraints. Toute relation d'équivalence sur les processus contraints est basée sur une comparaison qui doit prendre en considération les variables libres de chacun des processus SPPA, ainsi que des formules qui les contraint. Du coup, une telle comparaison doit nécessairement être accompagnée d'une relation qui détermine une correspondance entre les variables libres des deux processus.[11] Par exemple, pour affirmer que $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ sont équivalents, on doit absolument préciser, pour toute variable libre $x \in \mathrm{fv}(P)$, quelles variables libres $y \in \mathrm{fv}(Q)$ correspondent à x, et vice versa. Pour cette fin, nous utilisons des relations $R \in \mathfrak{R}$, avec $R \subseteq \mathrm{fv}(P) \times \mathrm{fv}(Q)$, introduites à la Section 4.2.

D'autre part, la notion de relation pleine est nécessaire à la définition d'une relation d'équivalence entre les processus contraints. En effet, en plus d'établir une correspondance entre les variables libres des processus comparés, nous devons nous assurer que la relation qui établit cette correspondance est pleine, c'est-à-dire que toute variable libre d'un processus est associée à au moins une variable de l'autre processus et vice versa.

4.6.1 Bisimulation

Pour la prochaine définition, rappelons qu'étant donnée une action de Type II $\beta = c_{id}(x)$, f_{id}, split_{id} ou dec_{id}, nous supposons que x est toujours la variable introduite par β.

Définition 4.39 (Bisimulation). Soient $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ des processus contraints et soit $R \in \mathfrak{R}$ une relation pleine entre $\mathrm{fv}(\phi)$ et $\mathrm{fv}(\psi)$. Une *bisimulation*

[11] Rappelons que lorsque nous comparons deux processus, nous supposons toujours qu'aucune variable n'est utilisée pour définir les deux processus à la fois.

entre $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ par rapport à R est une famille de relations $\mathcal{R} = \{\mathcal{R}^\varrho\}_\varrho$, pour toute évaluation ϱ, telle que chaque relation $\mathcal{R}^\varrho \subseteq \mathcal{D}(\langle P,\ \phi\rangle) \times \mathcal{D}(\langle Q,\ \psi\rangle) \times \mathfrak{R}$ satisfait les conditions suivantes :

1. Si $\varrho \models \phi$, $\varrho \models \psi$ et ϱ est cohérente avec R, alors $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^\varrho$;
2. Si $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R_1) \in \mathcal{R}^\varrho$, alors pour toute action $\alpha = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} ou τ, et toute action $\beta = c_{id}(x)$, f_{id}, split_{id} ou dec_{id}, nous avons
   - si $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha} \langle P_2,\ \phi_2\rangle$ et $\varrho \models \phi_2$, alors $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^\varrho$, où Q_2 et ψ_2 sont tels que $\varrho \models \psi_2$ et $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha'} \langle Q_2,\ \psi_2\rangle$, et α' est telle que $\alpha' = \alpha[y_1/x_1] \ldots [y_n/x_n]$ avec $(x_i, y_i) \in R_1$;
   - si $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha} \langle Q_2,\ \psi_2\rangle$ et $\varrho \models \psi_2$, alors $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^\varrho$ où P_2 et ϕ_2 sont tels que $\varrho \models \phi_2$ et $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha'} \langle P_2,\ \phi_2\rangle$, et $\alpha' = \alpha[y_1/x_1] \ldots [y_n/x_n]$ avec $(y_i, x_i) \in R_1$;
   - si $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta} \langle P_2,\ \phi_2\rangle$, alors $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{\varrho'[x/y]}$ pour toute évaluation ϱ' cohérente avec $R_1[x]$ telle que $\varrho' \models \phi_2$, où Q_2 et ψ_2 sont tels que $\varrho'[x/y] \models \psi_2$ et $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta'} \langle Q_2,\ \psi_2\rangle$, et $\beta' = \beta[y/x]$;
   - si $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta} \langle Q_2,\ \psi_2\rangle$, alors $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{\varrho'[x/y]}$ pour toute évaluation ϱ' cohérente avec $R_1[x]$ telle que $\varrho' \models \psi_2$, où P_2 et ϕ_2 sont tels que $\varrho'[x/y] \models \phi_2$ et $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta'} \langle P_2,\ \phi_2\rangle$, et $\beta' = \beta[y/x]$.

Les processus contraints $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ sont *bisimilaires* s'il existe une bisimulation entre $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ par rapport à une certaine relation pleine R entre $\mathrm{fv}(\phi)$ et $\mathrm{fv}(\psi)$. Dans ce cas, nous écrivons $\langle P,\ \phi\rangle \simeq \langle Q,\ \psi\rangle$.

Notons que si $\mathcal{R} = \{\mathcal{R}^\varrho\}_\varrho$ est une bisimulation, alors $\varrho \models \phi$ et $\varrho \models \psi$ dès que $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^\varrho$. De plus, ϱ est nécessairement cohérente avec la relation R.

Définition 4.40. Soit $\mathcal{O}$ un critère d'observation. Les processus contraints $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ sont *$\mathcal{O}$-bisimilaires* si $\langle P/\mathcal{O},\ \phi\rangle \simeq \langle Q/\mathcal{O},\ \psi\rangle$. Dans ce cas, nous écrivons $\langle P,\ \phi\rangle \simeq_\mathcal{O} \langle Q,\ \psi\rangle$.

Exemple 4.41. Considérons les processus contraints $\langle P,\ \mathbf{1}\rangle$ et $\langle Q,\ \mathbf{1}\rangle$ avec

$$\begin{aligned} P &::= c(x).\,\mathbf{0} \\ Q &::= c(y_1).\,\mathbf{0} \;+\; c(y_2).\,\mathbf{0}\;. \end{aligned}$$

On a

$$\langle P,\ \mathbf{1}\rangle \xrightarrow{c(x)} \langle \mathbf{0},\ \mathcal{M}(x)\rangle$$

et

$$\langle Q,\ \mathbf{1}\rangle \xrightarrow{c(y_1)} \langle \mathbf{0},\ \mathcal{M}(y_1)\rangle \quad \text{et} \quad \langle Q,\ \mathbf{1}\rangle \xrightarrow{c(y_2)} \langle \mathbf{0},\ \mathcal{M}(y_2)\rangle.$$

Nous pouvons constater que $\langle P,\ \mathbf{1}\rangle \simeq \langle Q,\ \mathbf{1}\rangle$. En effet, nous pouvons construire une bisimulation $\mathcal{R}$ entre $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ par rapport à la relation vide $\emptyset$ (car $\mathrm{fv}(\mathbf{1}) = \emptyset$).[12] Une telle bisimulation est donnée par la famille $\mathcal{R} = \{\mathcal{R}^\varrho\}_\varrho$ avec

$$\begin{aligned} &(\langle P,\ \mathbf{1}\rangle,\ \langle Q,\ \mathbf{1}\rangle,\ \emptyset) \in \mathcal{R}^\varrho && \text{pour tout } \varrho \\ &(\langle \mathbf{0},\ \mathcal{M}(x)\rangle,\ \langle \mathbf{0},\ \mathcal{M}(y_1)\rangle,\ R_1) \in \mathcal{R}^\varrho && \text{pour tout } \varrho \text{ telle que } \varrho(x) = \varrho(y_1) \\ &(\langle \mathbf{0},\ \mathcal{M}(x)\rangle,\ \langle \mathbf{0},\ \mathcal{M}(y_2)\rangle,\ R_2) \in \mathcal{R}^\varrho && \text{pour tout } \varrho \text{ telle que } \varrho(x) = \varrho(y_2) \end{aligned}$$

où $R_1 = \{(x, y_1)\}$ et $R_2 = \{(x, y_2)\}$.

4.6.2 Correspondance des bisimulations

Le prochain théorème affirme que l'équivalence de bisimulation entre les processus contraints SPPA (Définition 4.39) correspond à l'équivalence de bisimulation entre les processus SPPA (Définition 3.18). Évidemment, ce résultat est valide seulement lorsque les processus SPPA comparés sont compatibles, donc seulement pour les processus SPPA clos.

Théorème 4.42. *Soient P des Q processus clos. Alors, $P \simeq Q$ si et seulement si $\langle P,\ \mathbf{1}\rangle \simeq \langle Q,\ \mathbf{1}\rangle$.*

Démonstration. D'abord, supposons que $P \simeq Q$ et soit $\mathcal{R}$ une bisimulation entre P et Q. Considérons la famille de relations $\mathcal{R}' = \{\mathcal{R}'^\varrho\}_\varrho$, où, étant donnée une évaluation ϱ, la relation $\mathcal{R}'^\varrho$ est définie par :

- pour tout $(P', Q') \in \mathcal{R}$, nous posons $(\langle P',\ \mathbf{1}\rangle, \langle Q',\ \mathbf{1}\rangle, \emptyset) \in \mathcal{R}'^\varrho$;

[12] La relation vide est pleine par convention.

- pour tout $(P'[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n], Q'[\varrho(y_1)/y_1]\ldots[\varrho(y_m)/y_m]) \in \mathcal{R}$, pour toutes formules ϕ et ψ telles que $\mathrm{fv}(\phi) = \{x_1, \ldots, x_n\}$ et $\mathrm{fv}(\psi) = \{y_1, \ldots, y_m\}$, et pour toute relation $R \subseteq \mathrm{fv}(\phi) \times \mathrm{fv}(\psi)$, nous posons $(\langle P',\ \phi\rangle, \langle Q',\ \psi\rangle, R) \in \mathcal{R}'^{\varrho}$ lorsque $\varrho \models \phi$, $\varrho \models \psi$ et ϱ est cohérente avec R.

Dans ce qui suit, nous démontrons que la relation $\mathcal{R}' = \{\mathcal{R}'^{\varrho}\}_{\varrho}$ est une bisimulation entre $\langle P,\ \mathbf{1}\rangle$ et $\langle Q,\ \mathbf{1}\rangle$ par rapport à la relation vide $R = \emptyset$. Afin de démontrer cet énoncé, il suffit de voir que chaque relation $\mathcal{R}'^{\varrho}$ satisfait les conditions de la Définition 4.39.

1. D'abord, nous voyons que $(\langle P,\ \mathbf{1}\rangle, \langle Q,\ \mathbf{1}\rangle, \emptyset) \in \mathcal{R}'^{\varrho}$ (car $\varrho \models \mathbf{1}$ et ϱ est cohérente avec la relation vide $\emptyset$).

2. Soit $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R) \in \mathcal{R}'^{\varrho}$. Nous commençons par constater que $\varrho \models \phi_1$, $\varrho \models \psi_1$ et ϱ est cohérente avec R. De plus, nous pouvons supposer que $\mathrm{fv}(\phi_1) = \{x_1, \ldots, x_n\}$ et $\mathrm{fv}(\psi_1) = \{y_1, \ldots, y_m\}$, avec $R \subseteq \mathrm{fv}(\phi_1) \times \mathrm{fv}(\psi_1)$. D'autre part, nous avons $(P'_1, Q'_1) \in \mathcal{R}$, avec $P'_1 ::= P_1[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]$ et $Q'_1 ::= Q_1[\varrho(y_1)/y_1]\ldots[\varrho(y_m)/y_m]$.
   Supposons que $\varrho \models \phi_2$ et $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha'} \langle P_2,\ \phi_2\rangle$ avec $\alpha' = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} ou τ. Par le Lemme 4.37,

$$P'_1 \xrightarrow{\alpha} P'_2$$

   où $P'_2 ::= P_2[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]$ et $\alpha = \varrho(\alpha')$. Puisque $(P'_1, Q'_1) \in \mathcal{R}$, il existe une transition

$$Q'_1 \xrightarrow{\alpha} Q'_2$$

   telle que $(P'_2, Q'_2) \in \mathcal{R}$, où $Q'_2 ::= Q_2[\varrho(y_1)/y_1]\ldots[\varrho(y_m)/y_m]$. Ensuite, par le Lemme 4.36, il existe une transition

$$\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha''} \langle Q_2,\ \psi_2\rangle$$

   avec $\alpha = \alpha''[\varrho(y_1)/y_1]\ldots[\varrho(y_m)/y_m]$ et $\varrho \models \psi_2$ (par le Lemme 4.38). De plus, puisque ϱ est cohérente avec R, nous pouvons supposer que

$\alpha' = \alpha''[y'_1/x_1]\ldots[y'_n/x_n]$ avec $\{y'_1,\ldots,y'_n\} = \{y_1,\ldots,y_m\}$ et $(x_i, y'_i) \in R$. Ainsi, puisque $\varrho \models \phi_2$, $\varrho \models \psi_2$ et ϱ est cohérente avec R, nous pouvons constater que $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R) \in \mathcal{R}'^{\varrho}$. Le cas où nous avons une transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha'} \langle Q_2,\ \psi_2\rangle$ est similaire.

Maintenant, supposons que $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta'} \langle P_2,\ \phi_2\rangle$, avec $\beta' = c_{id}(x)$, f_{id}, split_{id} ou dec_{id}, et soit ϱ' une évaluation cohérente avec $R[x]$ telle que $\varrho' \models \phi_2$. Par le Lemme 4.37,

$$P'_1 \xrightarrow{\beta} P'_2$$

où $\beta = \varrho'(\beta')$ et $P'_2 ::= P_2[\varrho'(x)/x][\varrho'(x_1)/x_1]\ldots[\varrho'(x_n)/x_n]$. Puisque $(P'_1, Q'_1) \in \mathcal{R}$, il existe une transition

$$Q'_1 \xrightarrow{\beta} Q'_2$$

telle que $(P'_2, Q'_2) \in \mathcal{R}$, où $Q'_2 ::= Q_2[\varrho'(x)/y][\varrho'(y_1)/y_1]\ldots[\varrho'(y_m)/y_m]$ pour une certaine variable y. Par le Lemme 4.36, il existe une transition

$$\langle Q_1,\ \phi_1\rangle \xrightarrow{\beta''} \langle Q_2,\ \psi_2\rangle$$

avec $\beta = \beta''[\varrho'(x)/y] = \varrho'(\beta'')$ et $\varrho'[x/y] \models \psi_2$ (par le Lemme 4.38). Mais puisque ϱ' est cohérente avec $R[x]$, nous avons que $\beta' = \beta''[y/x]$. Ainsi, puisque $\varrho'[x/y] \models \phi_2$, $\varrho'[x/y] \models \psi_2$ et $\varrho'[x/y]$ est cohérente avec $R[\![(x,y)]\!]$, nous pouvons constater que $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R[\![(x,y)]\!]) \in \mathcal{R}'^{\varrho'[x/y]}$. Le cas où nous avons une transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta'} \langle Q_2,\ \psi_2\rangle$ est similaire.

Réciproquement, supposons que $\langle P,\ \mathbf{1}\rangle \simeq \langle Q,\ \mathbf{1}\rangle$ et soit $\mathcal{R} = \{\mathcal{R}^{\varrho}\}$ une bisimulation entre $\langle P,\ \mathbf{1}\rangle$ et $\langle Q,\ \mathbf{1}\rangle$ par rapport à la relation vide $\emptyset$. Considérons la relation $\mathcal{R}' \subseteq \mathcal{D}(P) \times \mathcal{D}(Q)$ définie comme suit :

- Si $(\langle P',\ \mathbf{1}\rangle,\ \langle Q',\ \mathbf{1}\rangle,\ \emptyset) \in \mathcal{R}^{\varrho}$ pour une certaine évaluation ϱ, alors $(P', Q') \in \mathcal{R}'$;

– Si $(\langle P',\ \phi\rangle,\ \langle Q',\ \psi\rangle,\ R) \in \mathcal{R}^{\varrho}$ pour une certaine évaluation ϱ, alors

$$(P'[\varrho(x_1)/x_1]\ldots[\varrho(x_1)/x_n],\ Q'[\varrho(y_1)/y_1]\ldots[\varrho(y_m)/y_m]) \in \mathcal{R}'$$

où $\mathrm{fv}(\phi) = \{x_1,\ldots,x_n\}$ et $\mathrm{fv}(\psi) = \{y_1,\ldots,y_m\}$.

Ainsi, il suffit de démontrer que la relation $\mathcal{R}'$ est une bisimulation entre les processus P et Q.

1. D'abord, nous constatons que $(P,Q) \in \mathcal{R}'$ car $(\langle P,\ \mathbf{1}\rangle,\ \langle Q,\ \mathbf{1}\rangle,\emptyset) \in \mathcal{R}^{\varrho}$, pour toute évaluation ϱ.

2. Soient $P_1' ::= P_1[a_1/x_1]\ldots[a_n/x_n]$ et $Q_1' ::= Q_1[b_1/y_1]\ldots[b_m/y_m]$, pour certains messages $a_i, b_j \in \mathcal{M}$, tels que $(P_1', Q_1') \in \mathcal{R}'$. Par la définition de $\mathcal{R}'$, il existe une évaluation ϱ, avec $\varrho(x_i) = a_i$ et $\varrho(y_j) = b_j$, telle que $(\langle P_1,\ \phi_1\rangle,\ \langle Q_1,\ \psi_1\rangle,\ R) \in \mathcal{R}^{\varrho}$ avec $\mathrm{fv}(\phi_1) = \{x_1,\ldots,x_n\}$ et $\mathrm{fv}(\psi_1) = \{y_1,\ldots,y_m\}$. De plus, nous avons $\varrho \models \phi_1$, $\varrho \models \psi_1$ et ϱ est cohérente avec la relation R.

   Soit $\alpha' = \overline{c}(t)$, $\delta^{c}_{id}(t)$, $\overline{\delta^{c}_{id}}(t)$, signv_{id} ou τ, et supposons que $P_1' \xrightarrow{\alpha} P_2'$ avec $\alpha = \alpha'[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n] = \varrho(\alpha)$ et $P_2' ::= P_2[\varrho(x_1)/x_1]\ldots[\varrho(x_n)/x_n]$. Par le Lemme 4.36 et puisque $\mathrm{fv}(\phi_1) = \{x_1,\ldots,x_n\}$, nous avons

$$\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha'} \langle P_2,\ \phi_2\rangle$$

   avec $\varrho \models \phi_2$ (par le Lemme 4.38). Mais puisque $(\langle P_1,\ \phi_1\rangle,\ \langle Q_1,\ \psi_1\rangle,\ R) \in \mathcal{R}^{\varrho}$, il existe une transition

$$\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha''} \langle Q_2,\ \psi_2\rangle$$

   pour $\alpha' = \alpha[y_1'/x_1]\ldots[y_n'/x_n]$ avec $\{y_1',\ldots,y_n'\} = \{y_1,\ldots,y_m\}$ et $(x_i, y_i') \in R$, telle que $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R) \in \mathcal{R}^{\varrho}$ et $\varrho \models \psi_2$. D'autre part, puisque ϱ est cohérente avec R, nous avons $\varrho(\alpha'') = \varrho(\alpha') = \alpha$. Ainsi,

par le Lemme 4.37 et puisque $\varrho \models \psi_2$, nous avons

$$Q'_1 \xrightarrow{\alpha} Q'_2$$

où $Q'_2 ::= Q_2[\varrho(y_1)/y_1] \dots [\varrho(y_m)/y_m]$. De plus, puisque $\mathrm{fv}(\phi_2) = \mathrm{fv}(\phi_1) = \{x_1, \dots, x_n\}$ et $\mathrm{fv}(\psi_2) = \mathrm{fv}(\psi_1) = \{y_1, \dots, y_m\}$, et puisque que nous avons $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R) \in \mathcal{R}^{\varrho}$, nous pouvons constater, directement de la définition de la relation $\mathcal{R}'$, que $(P'_2, Q'_2) \in \mathcal{R}'$. Le cas où nous avons une transition $Q'_1 \xrightarrow{\alpha} Q'_2$ est similaire.

Supposons maintenant que $P'_1 \xrightarrow{\beta} P'_2$ où $\beta = \beta'[a/x]$ pour une certaine action $\beta' = c_{id}(x)$, f_{id}, split_{id} ou dec_{id}, et pour un certain processus $P'_2 ::= P_2[a/x][\varrho(x_1)/x_1] \dots [\varrho(x_n)/x_n]$. Considérons l'évaluation ϱ' définie par :

$$\varrho'(x) = a \quad \text{et} \quad \varrho'(z) = \varrho(z) \text{ sinon.}$$

Puisque ϱ est cohérente avec R, l'évaluation ϱ' est nécessairement cohérente avec $R[x]$, donc $\beta = \varrho'(\beta')$. De plus, nous pouvons constater que $P'_2 = P_2[\varrho'(x)/x][\varrho'(x_1)/x_1] \dots [\varrho'(x_n)/x_n]$. Par le Lemme 4.36, et puisque $\mathrm{fv}(\phi_1) = \{x_1, \dots, x_n\}$, nous avons

$$\langle P_1,\ \phi_1\rangle \xrightarrow{\beta'} \langle P_2,\ \phi_2\rangle$$

avec $\varrho' \models \phi_2$ (par le Lemme 4.38). Mais puisque $(\langle P_1,\ \phi_1\rangle,\ \langle Q_1,\ \psi_1\rangle,\ R) \in \mathcal{R}^{\varrho}$, il existe une transition

$$\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta''} \langle Q_2,\ \psi_2\rangle$$

avec $\beta'' = \beta'[y/x]$, telle que $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R[\![(x,y)]\!]) \in \mathcal{R}^{\varrho'[x/y]}$ et $\varrho'[x/y] \models \psi_2$. D'autre part, puisque ϱ' est cohérente avec $R[x]$, donc ϱ' est cohérente avec $R[\![(x,y)]\!]$, nous avons $\varrho'[x/y](\beta'') = \varrho'[x/y](\beta') = \beta$. Posons

$\varrho'' = \varrho'[x/y]$. Par le Lemme 4.37 et puisque $\varrho'' \models \psi_2$, nous avons

$$Q_1' \xrightarrow{\beta} Q_2'$$

où $Q_2' ::= Q_2[\varrho''(y)/y][\varrho''(y_1)/y_1] \ldots [\varrho''(y_m)/y_m]$ avec

$$\varrho''(y) = a,\ \varrho''(y_1) = \varrho(y_1), \ldots, \varrho''(y_m) = \varrho(y_m).$$

De plus, puisque $\mathrm{fv}(\phi_2) = \mathrm{fv}(\phi_1) \cup \{x\} = \{x_1, \ldots, x_n, x\}$ et $\mathrm{fv}(\psi_2) = \mathrm{fv}(\psi_1) \cup \{y\} = \{y_1, \ldots, y_m, y\}$, et puisque $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R[\![(x,y)]\!]) \in \mathcal{R}^{\varrho''}$, nous pouvons constater, directement de la définition de $\mathcal{R}'$, que $(P_2', Q_2') \in \mathcal{R}'$. Le cas où nous avons une transition $Q_1' \xrightarrow{\beta} Q_2'$ est similaire.

□

4.7 Bisimulation symbolique

Dans cette section, nous introduisons une relation de bisimulation symbolique pour les processus contraints qui se décide en nombre fini d'étapes. Nous montrons également que cette relation de bisimulation symbolique est équivalente à la relation de bisimulation présentée à la Définition 4.39. Notre bisimulation symbolique peut donc nous servir de méthode de preuve cohérente et complète afin de décider de la bisimilarité de deux processus contraints.

Étant donnés deux processus contraints $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$, nous considérons les ensembles de formules suivants :

$$E_1 = \{\phi' \mid \langle P',\ \phi'\rangle \in \mathcal{D}(\langle P,\ \phi\rangle) \text{ pour un certain } P'\} = \{\phi_1, \ldots, \phi_m\}$$

et

$$E_2 = \{\psi' \mid \langle Q',\ \psi'\rangle \in \mathcal{D}(\langle Q,\ \psi\rangle) \text{ pour un certain } Q'\} = \{\psi_1, \ldots, \psi_{m'}\}$$

tels que

$$\bigcup_{1\leq j\leq m} \mathrm{fv}(\phi_j) = \{x_1,\ldots,x_n\} \quad \text{et} \quad \bigcup_{1\leq j\leq m'} \mathrm{fv}(\psi_j) = \{y_1,\ldots,y_{n'}\}.$$

Les ensembles de formules E_1 et E_2 sont nécessairement finis (modulo la $\mathcal{V}$-équivalence de formules $\overset{\mathcal{V}}{\Leftrightarrow}$) puisque les ensembles $\mathcal{D}(\langle P,\ \phi\rangle)$ et $\mathcal{D}(\langle Q,\ \psi\rangle)$ sont finis par le Théorème 4.34.

La relation de bisimulation symbolique que nous donnons ci-dessous est fondée sur la relation d'équivalence $\equiv_{E_1E_2}$ sur les évaluations (voir Définition 4.15). Pour les besions de cette section, nous désignons par $[\![\varrho]\!]$ la classe d'équivalence $[\![\varrho]\!]_{E_1E_2}$ par rapport à cette relation. Rappelons que la relation $\equiv_{E_1E_2}$ ne possède qu'un nombre fini de classes d'équivalences. De plus, ces classes d'équivalences sont préservées par rapport à la propriété de cohérence (selon une relation) et à l'opérateur de substitution de variable.

Définition 4.43 (Bisimulation symbolique). Soient $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ des processus contraints et soit $R \in \mathfrak{R}$ une relation pleine entre $\mathrm{fv}(\phi)$ et $\mathrm{fv}(\psi)$. Une *bisimulation symbolique* entre $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ par rapport à R est une famille finie $\mathcal{R} = \{\mathcal{R}^{[\![\varrho]\!]}\}_{\varrho}$, pour toute évaluation ϱ, telle que chaque relation $\mathcal{R}^{[\![\varrho]\!]}$ satisfait les conditions suivantes :

1. Si $\varrho \models \phi$, $\varrho \models \psi$ et ϱ est cohérente avec R, alors $(\langle P,\ \phi\rangle,\ \langle Q,\ \psi\rangle,\ R) \in \mathcal{R}^{[\![\varrho]\!]}$;
2. Si $(\langle P_1,\ \phi_1\rangle,\ \langle Q_1,\ \psi_1\rangle,\ R_1) \in \mathcal{R}^{[\![\varrho]\!]}$, alors pour toute action $\alpha = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} ou τ, et pour toute action $\beta = c_{id}(x)$, f_{id}, split_{id} ou dec_{id}, nous avons
   - si $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha} \langle P_2,\ \phi_2\rangle$ et $\varrho \models \phi_2$, alors $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^{[\![\varrho]\!]}$, où Q_2 et ψ_2 sont tels que $\varrho \models \psi_2$ et $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha'} \langle Q_2,\ \psi_2\rangle$, et $\alpha' = \alpha[y_1/x_1]\ldots[y_n/x_n]$ avec $(x_i, y_i) \in R_1$;
   - si $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha} \langle Q_2,\ \psi_2\rangle$ et $\varrho \models \psi_2$, alors $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^{[\![\varrho]\!]}$, où P_2 et ϕ_2 sont tels que $\varrho \models \phi_2$ et $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha'} \langle P_2,\ \phi_2\rangle$, et $\alpha' = \alpha[y_1/x_1]\ldots[y_n/x_n]$ avec $(y_i, x_i) \in R_1$;

- si $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta} \langle P_2,\ \phi_2\rangle$, alors $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{[\![\varrho'[x/y]]\!]}$ pour toute évaluation ϱ' cohérente avec $R_1[x]$ telle que $\varrho' \models \phi_2$, où Q_2 et ψ_2 sont tels que $\varrho'[x/y] \models \psi_2$ et $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta'} \langle Q_2,\ \psi_2\rangle$, et $\beta' = \beta[y/x]$;
- si $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta} \langle Q_2,\ \psi_2\rangle$, alors $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{[\![\varrho'[x/y]]\!]}$ pour toute évaluation ϱ' cohérente avec $R_1[x]$ telle que $\varrho' \models \psi_2$, où P_2 et ϕ_2 sont tels que $\varrho'[x/y] \models \phi_2$ et $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta'} \langle P_2,\ \phi_2\rangle$, et $\beta' = \beta[y/x]$.

Nous écrivons $\langle P,\ \phi\rangle \simeq_{\mathrm{s}} \langle Q,\ \psi\rangle$ si une telle bisimulation symbolique existe entre $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ par rapport à une certaine relation pleine R entre $\mathrm{fv}(\phi)$ et $\mathrm{fv}(\psi)$.

Le théorème suivant nous permet d'affirmer que la bisimulation symbolique constitue une méthode de preuve cohérente et complète pour la vérification de la bisimilarité entre deux processus contraints.

Théorème 4.44. *$\langle P,\ \phi\rangle \simeq \langle Q,\ \psi\rangle$ si et seulement si $\langle P,\ \phi\rangle \simeq_{\mathrm{s}} \langle Q,\ \psi\rangle$.*

Démonstration. D'abord, supposons que $\langle P,\ \phi\rangle \simeq \langle Q,\ \psi\rangle$, et soit $\mathcal{R} = \{\mathcal{R}^{\varrho}\}_{\varrho}$ une bisimulation par rapport à une certaine relation pleine R entre $\mathrm{fv}(\phi)$ et $\mathrm{fv}(\psi)$. Pour chaque classe d'équivalence $[\![\varrho]\!]$, nous considérons la relation

$$\mathcal{R}'^{[\![\varrho]\!]} = \bigcup_{\varrho' \in [\![\varrho]\!]} \mathcal{R}^{\varrho'}.$$

En particulier, nous avons $\mathcal{R}^{\varrho} \subseteq \mathcal{R}'^{[\![\varrho]\!]}$, pour toute évaluation ϱ. Il suffit de démontrer que la famille (finie) $\mathcal{R}' = \{\mathcal{R}'^{[\![\varrho]\!]}\}_{\varrho}$ est une bisimulation symbolique entre $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ par rapport à R, donc $\langle P,\ \phi\rangle \simeq_{\mathrm{s}} \langle Q,\ \psi\rangle$. En effet, étant donnée une classe d'équivalence $[\![\varrho]\!]$, nous pouvons constater que $\mathcal{R}'^{[\![\varrho]\!]}$ satisfait les conditions de la Définition 4.43.

1. Si $\varrho \models \phi$, $\varrho \models \psi$, et ϱ est cohérente avec R, alors $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^{\varrho}$, d'où $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^{[\![\varrho]\!]}$.
2. Supposons que $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R_1) \in \mathcal{R}^{[\![\varrho]\!]}$, avec $\varrho \models \phi_1$ et $\varrho \models \psi_1$. Alors nous avons $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R_1) \in \mathcal{R}^{\varrho'}$, pour une certaine évaluation

$\varrho' \equiv_{E_1 E_2} \varrho$, avec $\varrho' \models \phi_1$ et $\varrho' \models \psi_1$. Ainsi, si $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha} \langle P_2,\ \phi_2\rangle$ pour $\alpha = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} ou τ, et $\varrho \models \phi_2$, donc $\varrho' \models \phi_2$, alors il existe une transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha'} \langle Q_2,\ \psi_2\rangle$, pour une certaine action $\alpha' = \alpha[y_1/x_1]\ldots[y_n/x_n]$, avec $(x_i, y_i) \in R_1$, telle que $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^{\varrho'}$ et $\varrho' \models \psi_2$. Nous avons donc $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^{[\![\varrho']\!]} = \mathcal{R}^{[\![\varrho]\!]}$ et $\varrho \models \psi_2$. Le cas où nous avons une transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha} \langle Q_2,\ \psi_2\rangle$ est similaire.

Supposons maintenant que $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta} \langle P_2,\ \phi_2\rangle$, pour $\beta = c_{id}(x)$, f_{id}, split_{id} ou dec_{id}, et soit ϱ_1 une évaluation cohérente avec $R_1[x]$ et telle que $\varrho_1 \models \phi_2$. Alors il existe une transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta'} \langle Q_2,\ \psi_2\rangle$, pour une certaine action $\beta' = \beta[y/x]$, telle que $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{\varrho_1[x/y]}$ et $\varrho_1[x/y] \models \psi_2$. De plus, puisque $\mathcal{R}^{\varrho_1[x/y]} \subseteq \mathcal{R}^{[\![\varrho_1[x/y]]\!]}$, nous pouvons constater que $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{[\![\varrho_1[x/y]]\!]}$. Le cas où nous avons une transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta} \langle Q_2,\ \psi_2\rangle$ est similaire.

Réciproquement, supposons que $\langle P,\ \phi\rangle \simeq_{\mathrm{s}} \langle Q,\ \psi\rangle$ et soit $\mathcal{R} = \{\mathcal{R}^{[\![\varrho]\!]}\}_\varrho$ une bisimulation symbolique entre $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ par rapport à une certaine relation pleine R entre $\mathrm{fv}(\phi)$ et $\mathrm{fv}(\psi)$. Considérons la famille $\mathcal{R}' = \{\mathcal{R}^\varrho\}_\varrho$ où $\mathcal{R}^\varrho = \mathcal{R}^{[\![\varrho]\!]}$. Il suffit de constater que $\mathcal{R}'$ est une bisimulation entre $\langle P,\ \phi\rangle$ et $\langle Q,\ \psi\rangle$ par rapport à R, donc $\langle P,\ \phi\rangle \simeq \langle Q,\ \psi\rangle$. En effet, étant donnée une évaluation ϱ, nous démontrons que la relation $\mathcal{R}^\varrho = \mathcal{R}^{[\![\varrho]\!]}$ satisfait les conditions de la Définition 4.39.

1. Si $\varrho \models \phi$, $\varrho \models \psi$, et ϱ est cohérente avec R, alors $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^{[\![\varrho]\!]}$, d'où $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R) \in \mathcal{R}^\varrho$.

2. Supposons que $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R_1) \in \mathcal{R}^\varrho$, avec $\varrho \models \phi_1$ et $\varrho \models \psi_1$. Alors, nous avons $(\langle P_1,\ \phi_1\rangle, \langle Q_1,\ \psi_1\rangle, R_1) \in \mathcal{R}^{[\![\varrho]\!]}$. Par conséquent, si $\varrho \models \phi_2$ et $\langle P_1,\ \phi_1\rangle \xrightarrow{\alpha} \langle P_2,\ \phi_2\rangle$, pour $\alpha = \overline{c_{id}}(t)$, $\delta^c_{id}(t)$, $\overline{\delta^c_{id}}(t)$, signv_{id} ou τ, alors il existe une transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha'} \langle Q_2,\ \psi_2\rangle$, pour une certaine action $\alpha' = \alpha[y_1/x_1]\ldots[y_n/x_n]$, avec $(x_i, y_i) \in R_1$, telle que $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^{[\![\varrho]\!]}$ et $\varrho \models \psi_2$. D'où, $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1) \in \mathcal{R}^\varrho$. Le cas où nous avons une transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\alpha} \langle Q_2,\ \psi_2\rangle$ est similaire.

   Supposons maintenant que $\langle P_1,\ \phi_1\rangle \xrightarrow{\beta} \langle P_2,\ \phi_2\rangle$, pour $\beta = c_{id}(x)$, f_{id}, split_{id}

ou dec_{id}, et soit ϱ_1 une évaluation cohérente avec $R[x]$ et telle que $\varrho_1 \models \phi_2$. Alors, il existe une transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta'} \langle Q_2,\ \psi_2\rangle$, pour une certaine action $\beta' = \beta[y/x]$, telle que $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{[\![\varrho_1[x/y]]\!]}$ et $\varrho_1[x/y] \models \psi_2$. De plus, puisque $\mathcal{R}^{\varrho_1[x/y]} = \mathcal{R}^{[\![\varrho_1[x/y]]\!]}$, nous avons $(\langle P_2,\ \phi_2\rangle, \langle Q_2,\ \psi_2\rangle, R_1[\![(x,y)]\!]) \in \mathcal{R}^{\varrho_1[x/y]}$. Le cas où nous avons une transition $\langle Q_1,\ \psi_1\rangle \xrightarrow{\beta} \langle Q_2,\ \psi_2\rangle$ est similaire.

□

Le Théorème 4.44 nous permet donc de construire une bisimulation entre deux processus contraints à partir de seulement un nombre fini d'évaluations (plus précisément, une pour chaque classe d'équivalence $[\![\varrho]\!]$). Conséquemment, ce résultat nous procure une méthode de preuve finie afin de décider de la bisimilarité des processus contraints.

Exemple 4.45. Considérons les processus A et B définis comme suit :

$$A ::= c(x_1).A', \quad A' ::= c(x_2).A \quad \text{et} \quad B ::= c(y).B$$

avec $id_A = id_B = id$. La sémantique symbolique des processus contraints $\langle A,\ \mathbf{1}\rangle$ et $\langle B,\ \mathbf{1}\rangle$ est illustrée à la Figure 4.8. Dans cet exemple, nous démontrons que ces deux processus contraints sont bisimilaires. Par le Théorème 4.44, il suffit de construire une bisimulation symbolique entre $\langle A,\ \mathbf{1}\rangle$ et $\langle B,\ \mathbf{1}\rangle$ par rapport à la relation vide $R = \emptyset$. Du coup, nous pourrons conclure que les processus SPPA A et B sont bisimilaires (par le Théorème 4.42).

Tout d'abord, nous considérons les ensembles formés des formules apparaissant, respectivement, dans $\mathcal{D}(\langle A,\ \phi\rangle)$ et $\mathcal{D}(\langle B,\ \psi\rangle)$:

$$E_1 = \{\mathbf{1},\ \mathcal{M}(x_1),\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\} \quad \text{et} \quad E_2 = \{\mathbf{1},\ \mathcal{M}(y)\}.$$

De plus, il suffit de considérer les ensembles de variables $\{x_1, x_2\}$ et $\{y\}$. Puisque ces formules sont satisfaites par toute évaluation, il n'y a que cinq classes d'équivalence

$$\langle A,\ \mathbf{1}\rangle \xrightarrow{c_{id}(x_1)} \langle A',\ \mathcal{M}(x_1)\rangle \xrightarrow{c_{id}(x_2)} \langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$$
$$\Big\downarrow{\scriptstyle c_{id}(x_1)} \quad \Big\uparrow{\scriptstyle c_{id}(x_2)}$$
$$\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$$

$$\langle B,\ \mathbf{1}\rangle \xrightarrow{c_{id}(y)} \langle B,\ \mathcal{M}(y)\rangle \xleftarrow{c_{id}(y)}$$

FIG. 4.8 – Sémantique symbolique de $\langle A,\ \mathbf{1}\rangle$ et $\langle B,\ \mathbf{1}\rangle$.

par rapport à la relation $\equiv_{E_1E_2}$, à savoir celles qui couvrent tous les cas où les variables prennent exactement les mêmes valeurs ou des valeurs différentes :

$$\begin{aligned}
[\![\varrho_1]\!]_{E_1E_2} &= \{\varrho \mid \varrho(x_1) \neq \varrho(x_2),\ \varrho(x_1) \neq \varrho(y) \text{ et } \varrho(x_2) \neq \varrho(y)\},\\
[\![\varrho_2]\!]_{E_1E_2} &= \{\varrho \mid \varrho(x_1) = \varrho(x_2) \text{ et } \varrho(x_1) \neq \varrho(y)\},\\
[\![\varrho_3]\!]_{E_1E_2} &= \{\varrho \mid \varrho(x_1) = \varrho(y) \text{ et } \varrho(x_1) \neq \varrho(x_2)\},\\
[\![\varrho_4]\!]_{E_1E_2} &= \{\varrho \mid \varrho(x_1) \neq \varrho(x_2) \text{ et } \varrho(x_2) = \varrho(y)\}, \text{ et}\\
[\![\varrho_5]\!]_{E_1E_2} &= \{\varrho \mid \varrho(x_1) = \varrho(x_2) = \varrho(y)\}.
\end{aligned}$$

Notons que nous avons $[\![\varrho_i[x_1/y]]\!]_{E_1E_2} = [\![\varrho_i[y/x_1]]\!]_{E_1E_2} = [\![\varrho_3]\!]_{E_1E_2}$ pour $i \in \{1,3\}$, $[\![\varrho_i[x_1/y]]\!]_{E_1E_2} = [\![\varrho_i[y/x_1]]\!]_{E_1E_2} = [\![\varrho_5]\!]_{E_1E_2}$ pour $i \in \{2,4,5\}$, $[\![\varrho_i[x_2/y]]\!]_{E_1E_2} = [\![\varrho_i[y/x_2]]\!]_{E_1E_2} = [\![\varrho_4]\!]_{E_1E_2}$ pour $i \in \{1,4\}$, et $[\![\varrho_i[x_2/y]]\!]_{E_1E_2} = [\![\varrho_i[y/x_2]]\!]_{E_1E_2} = [\![\varrho_5]\!]_{E_1E_2}$ pour $i \in \{2,3,5\}$.

Pour toute classe d'équivalence $[\![\varrho_i]\!]_{E_1E_2}$, les étapes de la construction de la relation $\mathcal{R}^{[\![\varrho_i]\!]_{E_1E_2}}$ sont illustrées à la Figure 4.9. En bref, pour chaque classe $[\![\varrho_i]\!]_{E_1E_2}$, nous devons parcourir les conditions imposées par la Définition 4.43 et déterminer quels couples de dérivées de l'ensemble $\mathcal{D}(\langle A,\ \mathbf{1}\rangle) \times \mathcal{D}(\langle B,\ \mathbf{1}\rangle)$ doit appartenir à la relation $\mathcal{R}^{[\![\varrho_i]\!]_{E_1E_2}}$, de même que la relation R qui lie leurs variables libres. La procédure de construction de ces relations est présentée de façon plus détaillée dans ce qui suit.

Pour la classe $[\![\varrho_1]\!]_{E_1E_2}$: nous ajoutons d'abord le triplet $(\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)$ à la relation $\mathcal{R}^{[\![\varrho_1]\!]_{E_1E_2}}$. Ensuite, puisque le processus contraint $\langle A,\ \mathbf{1}\rangle$ peut exé-

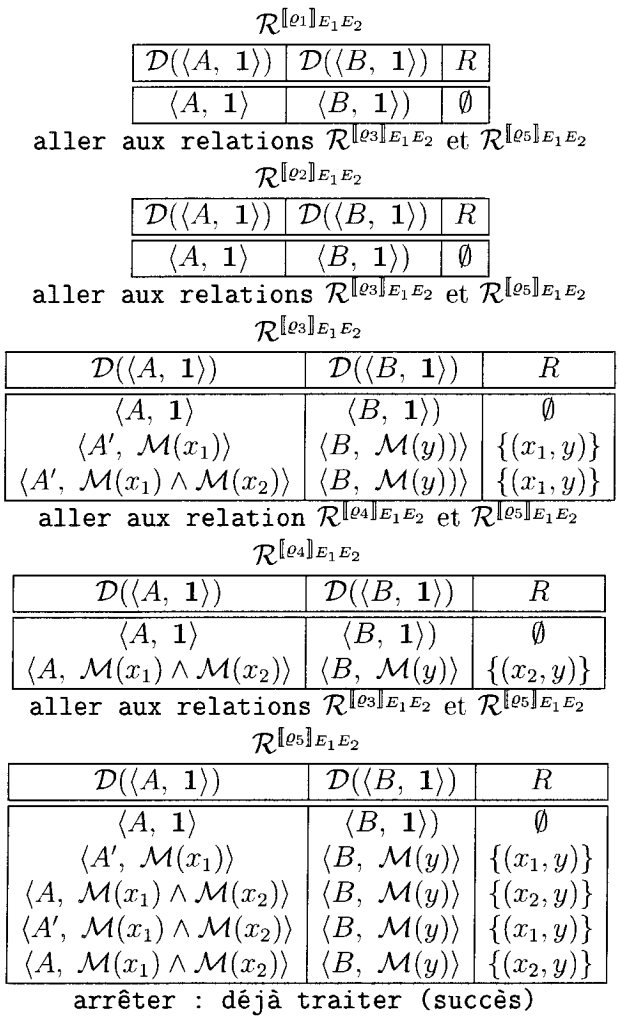

FIG. 4.9 – Bisimulation symbolique de $\langle A,\ \mathbf{1}\rangle$ et $\langle B,\ \mathbf{1}\rangle$.

cuter l'action d'entrée $c_{id}(x_1)$, nous devons vérifier que $\langle B,\ \mathbf{1}\rangle$ peut simuler cette action d'entrée avec sa propre action $c_{id}(y)$, où x_1 correspond à y. Mais puisque les évaluations ϱ_3 et ϱ_5 sont toutes deux cohérentes avec la relation $\{(x_1, y)\}$, nous devons ajouter $(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ à la relation $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$ et à la relation $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$. De façon analogue, la simulation de l'action d'entrée $c_{id}(y)$ de $\langle B,\ \mathbf{1}\rangle$ par l'action d'entrée $c_{id}(x_1)$ de $\langle A,\ \mathbf{1}\rangle$ nécessite le même ajout aux relations $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$ et $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$.

Pour la classe $[\![\varrho_2]\!]_{E_1E_2}$: Nous ajoutons d'abord $(\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)$ à $\mathcal{R}^{[\![\varrho_2]\!]_{E_1E_2}}$. Comme c'était le cas pour la classe $[\![\varrho_1]\!]_{E_1E_2}$, nous devons ensuite ajouter le triplet $(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ aux deux relations $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$ (déjà présent) et $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$ (déjà présent).

Pour la classe $[\![\varrho_3]\!]_{E_1E_2}$: Nous ajoutons d'abord $(\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)$ à $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$. Comme les autres classes ci-dessus, nous devons aussi ajouter le triplet $(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ aux relations $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$ (déjà présent) et $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$ (déjà présent). L'ajout de $(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ à la relation $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$ (imposé par les autres relations) nécessite à son tour la bisimulation de l'action d'entrée $c_{id}(x_2)$ provenant de $\langle A',\ \mathcal{M}(x_1)\rangle$ par l'action d'entrée $c_{id}(y)$ provenant de $\langle B,\ \mathcal{M}(y)\rangle$, par rapport à la relation $\{(x_2, y)\}$. D'où, puisque seules les évaluations ϱ_4 et ϱ_5 sont cohérentes avec la relation $\{(x_2, y)\}$, nous devons donc ajouter aux relations $\mathcal{R}^{[\![\varrho_4]\!]_{E_1E_2}}$ et $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$ le triplet $(\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\})$. D'autre part, l'ajout de $(\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ à $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$ (imposé par relation $\mathcal{R}^{[\![\varrho_4]\!]_{E_1E_2}}$) nécessite la bisimulation de l'action d'entrée $c_{id}(x_2)$ provenant de $\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$ par l'action d'entrée $c_{id}(y)$ provenant de $\langle B,\ \mathcal{M}(y)\rangle$, par rapport à la relation $\{(x_2, y)\}$. Nous devons donc ajouter le triplet $(\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\})$ aux relations $\mathcal{R}^{[\![\varrho_4]\!]_{E_1E_2}}$ (déjà présent) et $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$ (déjà présent).

Pour la classe $[\![\varrho_4]\!]_{E_1E_2}$: Nous ajoutons d'abord $(\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)$ à $\mathcal{R}^{[\![\varrho_4]\!]_{E_1E_2}}$. Comme les autres classes ci-dessus, nous devons ensuite ajouter le triplet

$(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ aux relations $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$ (déjà présent) et $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$ (déjà présent). L'ajout, imposé par la relation $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$, de $(\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\})$ à la relation $\mathcal{R}^{[\![\varrho_4]\!]_{E_1E_2}}$ nécessite la bisimulation de l'action d'entrée $c_{id}(x_1)$ provenant de $\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$ par l'action d'entrée $c_{id}(y)$ provenant de $\langle B,\ \mathcal{M}(y)\rangle$, par rapport à la relation $\{(x_1, y)\}$. Puisque les évaluations ϱ_3 et ϱ_5 sont cohérentes avec la relation $\{(x_1, y)\}$, nous devons ajouter $(\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ aux relations $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$ et $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$.

Pour la classe $[\![\varrho_5]\!]_{E_1E_2}$: Nous ajoutons d'abord $(\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)$ à $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$. Comme les autres classes ci-dessus, nous devons aussi ajouter le triplet $(\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})$ aux relations $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$ (déjà présent) et $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$ (déjà présent). Ce dernier ajout à $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$ nécessite la bisimulation de l'action d'entrée $c_{id}(x_2)$ provenant de $\langle A',\ \mathcal{M}(x_1)\rangle$ par l'action d'entrée $c_{id}(y)$ provenant de $\langle B,\ \mathcal{M}(y)\rangle$, par rapport à la relation $\{(x_2, y)\}$. Mais puisque les évaluations ϱ_4 et ϱ_5 sont cohérentes avec la relation $\{(x_2, y)\}$, nous devons ajouter le triplet $(\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\})$ aux relations $\mathcal{R}^{[\![\varrho_4]\!]_{E_1E_2}}$ (déjà présent) et $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$ (déjà présent). Cette dernier ajout à $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$ nécessite ensuite un bisimulation des processus contraints $\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle$ et $\langle B,\ \mathcal{M}(y)\rangle$, donc de leur action d'entrée respective $c_{id}(x_1)$ et $c_{id}(y)$ par rapport à la relation $\{(x_1, y)\}$. Mais puisque les évaluations ϱ_3 et ϱ_5 sont cohérentes avec la relation $\{(x_1, y)\}$, nous devons ajouter le triplet $(\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\}$ aux relations $\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}}$ (déjà présent) et $\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}}$ (déjà présent).

Cette procédure s'arrête (avec un succès) lorsque tout triplet $(\langle P,\ \phi\rangle, \langle Q,\ \psi\rangle, R)$ a été traité, sans aucune contradiction, pour toute relation $\mathcal{R}^{[\![\varrho]\!]_{E_1E_2}}$ telle que ϱ est cohérente avec R. Dans notre exemple, la procédure de construction de la bisimulation s'arrête car tous ces triplets ont été ajoutés, sans obtenir de contradiction, lorsque prescrit. Nous obtenons ainsi la bisimulation symbolique $\mathcal{R} = \bigcup_{i=1}^{5} \{\mathcal{R}^{[\![\varrho_i]\!]_{E_1E_2}}\}$, avec

$$\mathcal{R}^{[\![\varrho_1]\!]_{E_1E_2}} = \{\ (\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)\ \}$$

$$\mathcal{R}^{[\![\varrho_2]\!]_{E_1E_2}} = \{\ (\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset)\ \}$$

$$\mathcal{R}^{[\![\varrho_3]\!]_{E_1E_2}} = \{\ (\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset),\ (\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\}),$$
$$(\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})\ \}$$

$$\mathcal{R}^{[\![\varrho_4]\!]_{E_1E_2}} = \{\ (\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset),\ (\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\})\ \}$$

$$\mathcal{R}^{[\![\varrho_5]\!]_{E_1E_2}} = \{\ (\langle A,\ \mathbf{1}\rangle, \langle B,\ \mathbf{1}\rangle, \emptyset),\ (\langle A',\ \mathcal{M}(x_1)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\}),$$
$$(\langle A,\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_2, y)\}),$$
$$(\langle A',\ \mathcal{M}(x_1) \wedge \mathcal{M}(x_2)\rangle, \langle B,\ \mathcal{M}(y)\rangle, \{(x_1, y)\})\ \}.$$

4.8 Discussion

Nous avons présenté dans ce chapitre un modèle de spécification symbolique pour l'analyse de protocoles de sécurité. Ce modèle est basé sur une algèbre de messages qui formalise les principales primitives cryptographiques et sur une logique dont les termes atomiques sont ceux de notre algèbre de messages. Nous avons introduis la notion de processus contraint qui correspond à un couple formé d'un processus SPPA et d'une formule. Ils permettent une spécification explicite de la génération de valeurs symboliques, tels des nombres aléatoires, des nonces frais et des messages forgés. Nous démontrons que la sémantique des processus contraints correspond à des graphes de transitions finis, et sur lesquels nous établissons une équivalence de bisimulation. En outre, nous démontrons que cette équivalence de bisimulation est une généralisation de l'équivalence de bisimulation pour les processus SPPA. De plus, nous présentons une méthode de preuve cohérente et complète, appelée bisimulation symbolique, qui permet de décider si deux processus contraints sont bisimilaires.

La principale différence entre notre approche et celle de Hennessy-Lin [57] réside dans le graphe de transition symbolique : dans notre modèle symbolique, nous assignons à chaque état (processus) une formule qui offre une description précise des variables présentes dans le processus, tandis que le modèle symbolique de Hennessy-Lin nécessite une formule obtenue en considérant tous les chemins menant à l'état

(processus). Notre modèle symbolique fut développé avec un objectif d'analyse de protocoles de sécurité en tête – il est donc essentiel d'avoir une description détaillée des valeurs symboliques à un moment précis du protocole afin d'augmenter l'efficacité de l'analyse. En effet, l'analyse de protocoles de sécurité requiert fréquemment une vérification des effets de l'introduction de valeurs aléatoires (par exemple, un message forgé envoyé par un intrus) sur certains participants du protocole. Dans ce contexte, notre notion de processus contraint nous permet d'identifier directement ces valeurs aléatoires qui pourraient mener à une attaque. Par exemple, l'analyse d'un protocole contre les attaques de DoS nécessite généralement de vérifier si un message forgé envoyé par un intrus peut engendrer l'exécution de fonctions coûteuses en terme de ressources (telles le décryptage de données ou la vérification d'un message signé). Dans ce cas, une stratégie d'analyse basée sur les processus contraints consiste à vérifier, pour tout processus contraint obtenu à la suite d'une telle action coûteuse, les restrictions imposées par formule sur la variable qui représente le message forgé : si tous les messages forgés satisfont la formule, alors nous pouvons conclure que le protocole est incapable de détecter et contrer une session frauduleuse ; si la formule est satisfaite seulement par un petit nombre de messages, alors nous pouvons conclure que le protocole est sécuritaire car la majorité des attaques initiées par un intrus auront été préalablement détectées. Une méthode similaire qui permet de détecter si un protocole est vulnérable face aux attaques de DoS est présentée au Chapitre 6.

De façon plus générale, les méthodes d'analyse de protocoles sécurité que nous présenterons dans les prochains chapitres sont essentiellement fondées sur une méthode d'*équivalence-checking*. L'idée principale derrière cette approche consiste à vérifier si le protocole se comporte toujours correctement lorsque confronté à un environnement hostile. Dans le contexte de notre modèle symbolique, étant donné un processus contraint $\langle P,\ \phi_P \rangle$ spécifiant le protocole, il suffit grossièrement de vérifier si le protocole attaqué, spécifié par le processus contraint $\langle P_E,\ \phi_{P_E} \rangle$, est équivalent au protocole non attaqué, c'est-à-dire $\langle P,\ \phi_P \rangle$. Nous utilions la relation de $\mathcal{O}$-bisimulation (Définition 4.40) afin de comparer ces deux processus contraints.

De façon générale, toute méthode d'*équivalence-checking* basée sur une équivalence de bisimulation se transfère dans notre modèle symbolique : il suffit de remplacer les processus comparés, disons P et Q, par les processus contraints correspondant, $\langle P,\ \mathbf{1}\rangle$ et $\langle Q,\ \mathbf{1}\rangle$. Ainsi, la méthode qui requiert la vérification $P \simeq Q$ devient celle où l'on doit vérifier $\langle P,\ \mathbf{1}\rangle \simeq \langle Q,\ \mathbf{1}\rangle$. De plus, le théorème 4.42 nous permet de conclure que la méthode obtenue équivaut la première. En outre, cette nouvelle formulation nous permet maintenant de vérifier des processus infinis. Du coup, nous obtenons un autre argument en faveur de l'utilisation d'une méthode d'*équivalence-checking* comparativement à une méthode de *model-checking*.

CHAPITRE 5

VALIDATION DE PROTOCOLES DE SÉCURITÉ

L'un des principaux objectifs dans le domaine de la validation de protocoles de sécurité consiste à développer des algorithmes, communément appelés méthodes de preuve, cohérents et complets qui permettent de vérifier automatiquement si un protocole satisfait une certaine propriété de sécurité. L'utilisation d'une méthode de *model-checking* nécessite généralement une logique à partir de laquelle nous obtenons une spécification de la propriété de sécurité. Cette dernière est ensuite vérifiée à l'aide d'une procédure de décision propre à la logique. Il y a une grande variété de logiques qui permettent la spécification de propriétés sur les graphes de transitions (automates) et celles-ci diffèrent les unes des autres essentiellement au niveau du type de propriétés qu'elles peuvent exprimer. (Plusieurs de ces logiques sont recensées par Arnold [12].) Le μ-calcul, vue comme une extension de la logique de Hennessy-Milner [58], est une logique qui permet d'exprimer les propriétés de sécurité en terme des traces d'un processus. Une classification des propriétés définissables dans le μ-calcul en terme de propriétés de *préservation* (*safety*) et de *vivacité* (*liveness*) fut proposée par Lamport [73]. Cependant, il s'avère que ces propriétés de préservation et de vivacité ne sont pas suffisamment expressives pour spécifier certaines propriétés de sécurité, tout particulièrement les propriétés de non interférence.

L'idée centrale derrière les méthodes basées sur la non interférence consiste, brièvement, à démontrer qu'aucun intrus ne peut interférer avec un protocole. Dans cette thèse, nous présentons un raffinement de ces méthodes obtenu en considérant certaines interférences comme étant admissibles, par exemple l'interférence causée par le cryptage de données. Ce type d'interférence admissible s'exprime simplement en identifiant les actions de déclassification qui correspondent aux actions de cryptage (appels de fonction) qui se produisent lors de l'exécution du protocole. Notre méthode, qui est basée sur l'interférence admissible, nous procure deux types

d'avantages sur les autres méthodes basées sur la non interférence. En effet, dans certains cas, notre méthode permet une analyse des flots d'information d'un protocole sans pour autant avoir à étendre notre algèbre de processus à l'aide de règles de déductions parallèles afin d'exprimer les manipulations cryptographiques. Dans d'autres cas, notre méthode nous permet de mettre au rebut certaines interférences inoffensives, c'est-à-dire qui ne correspondent pas à des attaques réussies, lors de la phase de spécification du protocole ; ceci nous évite d'effectuer un dépistage manuel des résultats de la procédure de vérification.

Dans ce chapitre, nous introduisons la propriété d'*interférence admissible non déterministe par bisimulation* (BNAI) qui admet une caractérisation algébrique basée sur le concept de $\mathcal{O}$-bisimulation. Nous démontrons également un résultat relatif à la compositionalité de BNAI par rapport aux principaux opérateurs de SPPA. D'autre part, nous démontrons que la propriété BNAI, de même que la plus part des propriétés de non interférence, ne sont pas définissables dans le μ-calcul. Ce résultat constitue donc un important argument en faveur de l'utilisation de méthodes d'*équivalence-checking* pour la vérification de ce type de propriétés de flots d'information. En effet, une alternative basée sur une méthode de *model-checking* requiert la lourde tâche de développer une nouvelle logique pour la spécification des propriétés de sécurité (car nous ne pouvons pas utiliser le μ-calcul) et de nouveaux algorithmes de vérification pour celles-ci. L'approche « *équivalence-checking* », pour laquelle nous avons optée dans cette thèse, nous permet donc de réutiliser le concept naturel de bisimulation afin de vérifier la propriété BNAI. De plus, cette approche, contrairement à une approche basée sur une nouvelle logique, s'étend très facilement à notre modèle symbolique.

La propriété BNAI fut initialement présentée dans [84]. Cette publication contient également une brève présentation de la méthode de preuve par décomposition associée à BNAI. La démonstration que BNAI n'est pas définissable dans le μ-calcul est originale à cette thèse.

5.1 Non interférence forte non déterministe par bisimulation

Étant donné un processus P et deux sous-ensembles disjoints K et L de l'ensemble Vis des actions visibles, nous disons que K cause de l'*interférence* sur L (dans le processus P) s'il existe des actions de K qui provoquent (dans P) l'occurrence d'une action provenant de L qui ne se serait pas produite autrement. Par exemple, de la Figure 5.1 nous pouvons constater que l'action α_1 cause de l'interférence sur l'action α_2 dans le processus Q, mais pas dans le processus P.

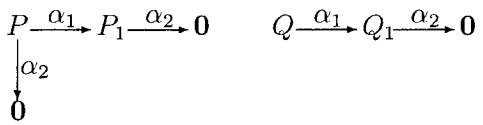

FIG. 5.1 – Illustration de l'interférence.

La prochaine formulation de la non interférence (par rapport aux ensembles L et K) exige qu'un processus soit $\mathcal{O}_L$-bisimulaire au processus obtenu en retirant les actions de K, de façon à interdire toute corrélation entre les comportements provenant de K et ceux provenant de L.

Définition 5.1 (Non interférence forte non déterministe par bisimulation). Le processus P satisfait BSNNI si

$$P \simeq_{\mathcal{O}_L} P \setminus K.$$

Cette définition nous procure une caractérisation algébrique de BSNNI basée sur la $\mathcal{O}_L$-bisimulation que nous pouvons facilement adopter comme méthode de vérification pour BSNNI.

Remarque 5.2. Il est important de noter que lorsque K correspond à l'ensemble Hi des actions visibles de haut-niveau et L correspond à l'ensemble Lo des actions visibles de bas-niveau, nous pouvons démontrer, à l'aide de la Proposition 3.20 et de

la Définition 5.1, que la propriété BSNNI coïncide avec la *bisimulation-based strong non-deterministic non-interference* telle que proposée par Focardi & Gorrieri [45].

Exemple 5.3. Considérons les processus P et Q dont la sémantique est illustrée à la Figure 5.1 (de la page 111), avec $\alpha_1 \in K$ et $\alpha_2 \in L$. Nous démontrons que le processus P satisfait BSNNI et que le processus Q ne satisfait pas BSNNI. Pour ce, nous devons déterminer si les processus P et $P\backslash K$ (respectivement Q et $Q\backslash K$) sont $\mathcal{O}_L$-bisimilaires. Considérons tout d'abord la Figure 5.2 qui contient la sémantique du processus $P\backslash K$, ainsi celles des processus $P/\mathcal{O}_L$ et $(P\backslash K)/\mathcal{O}_L$, observés selon le critère d'observation $\mathcal{O}_L$. (Notons que pour ces deux derniers processus, nous avons omit des transitions τ qui bouclent pour tout état du graphe.) Directement de la sémantique des processus $P/\mathcal{O}_L$ et $(P\backslash K)/\mathcal{O}_L$, nous pouvons conclure que P satisfait BSNNI car $P/\mathcal{O}_L \simeq (P \setminus K)/\mathcal{O}_L$, donc $P \simeq_{\mathcal{O}_L} P \setminus K$.

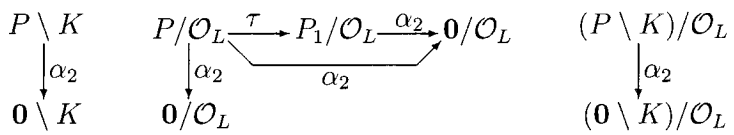

FIG. 5.2 – Sémantique des processus $P \setminus K$, $P/\mathcal{O}_L$ et $(P \setminus K)/\mathcal{O}_L$.

Considérons maintenant le processus Q. Nous commençons par constater que la sémantique de $Q \setminus K$ peut exécuter aucune transition, d'où $Q \setminus K \simeq \mathbf{0}$. Il en sera de même pour le processus observé $(Q \setminus K)/\mathcal{O}_L$, c'est-à-dire $(Q \setminus K)/\mathcal{O}_L \simeq \mathbf{0}$. D'autre part, de la sémantique du processus $Q/\mathcal{O}_L$ qui est illustrée à la Figure 5.3, nous pouvons clairement constater que $Q/\mathcal{O}_L \not\simeq \mathbf{0}$, d'où $Q/\mathcal{O}_L \not\simeq (Q \setminus K)/\mathcal{O}_L$ et $Q \not\simeq_{\mathcal{O}_L} Q \setminus K$. Par la Définition 5.1, nous pouvons donc conclure que le processus Q ne satisfait pas BSNNI.

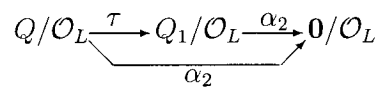

FIG. 5.3 – Sémantique du processus $(Q \setminus K)/\mathcal{O}_L$.

5.2 Interférence admissible non déterministe par bisimulation

Étant donné un ensemble $\Gamma \subseteq Vis$ d'actions de déclassification, l'interférence admissible fait référence à la propriété de flots d'information qui exige qu'un système admette des flots d'information provenant de comportements de K et qui influencent les comportements observables de L seulement à travers des actions de déclassification. Mullins [83] a proposé une formalisation de cette propriété qui exige que tout processus dérivé du processus initial et n'exécutant aucune action de déclassification doit satisfaire la non interférence. Plus précisément, un processus P satisfait la non interférence intransitive (voir Section 2.4.2), si les processus $P' \setminus \Gamma$ satisfont la non interférence pour toute dérivée $P' \in \mathcal{D}(P)$. Lorsque nous reformulons en utilisant BSNNI comme propriété de non interférence, nous obtenons la propriété d'*interférence admissible non déterministe par bisimulation* (BNAI).

Définition 5.4 (Interférence admissible non déterministe par bisimulation). Le processus P satisfait BNAI si

$$\forall_{P' \in \mathcal{D}(P)} \quad P' \setminus \Gamma \simeq_{\mathcal{O}_L} P' \setminus (\Gamma \cup K).$$

Grâce à cette caractérisation algébrique de BNAI basée sur la $\mathcal{O}_L$-bisimulation, la démonstration qu'un processus P satisfait BNAI se ramène à construire, pour chaque dérivée P', une $\mathcal{O}_L$-bisimulation entre $P' \setminus \Gamma$ et $P' \setminus (\Gamma \cup K)$. Lorsque le graphe de transitions associé au processus P est fini, cette procédure peut être accomplie automatiquement.

Exemple 5.5. Considérons le processus P dont la sémantique est illustrée à la Figure 5.4, avec $\alpha_1 \in K$, $\alpha_2 \in L$ et $\beta \in \Gamma$. Nous pouvons constater que le processus

P satisfait BNAI. Pour ce, nous considérons la sémantique du processus $P \setminus \Gamma$, et de ses dérivées, qui est illustrée à la Figure 5.5, et celle du processus $P \setminus (\Gamma \cup K)$, et de ses dérivées, qui est illustrée à la Figure 5.6. De ces deux Figures, nous pouvons constater que $P' \setminus \Gamma \simeq_{\mathcal{O}_L} P' \setminus (\Gamma \cup K)$ pour toute dérivée $P' \in \mathcal{D}(P) = \{P,\ P_1,\ P_2,\ P_3,\ \mathbf{0}\}$. En effet, la relation

$$R = \{((P' \setminus \Gamma)/\mathcal{O}_L,\ (P' \setminus (\Gamma \cup K))/\mathcal{O}_L) \mid P' \in \mathcal{D}(P)\}$$

est une bisimulation entre $(P' \setminus \Gamma)/\mathcal{O}_L$ et $(P' \setminus (\Gamma \cup K))/\mathcal{O}_L$, pour toute dérivée P'. Ainsi, par la Définition 5.4, nous pouvons conclure que le processus P satisfait BNAI.

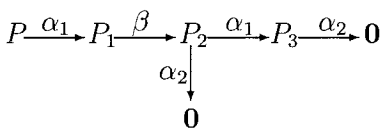

FIG. 5.4 – Sémantique du processus P.

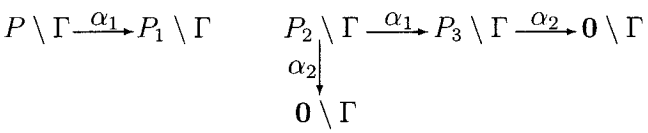

FIG. 5.5 – Sémantique du processus $P \setminus \Gamma$.

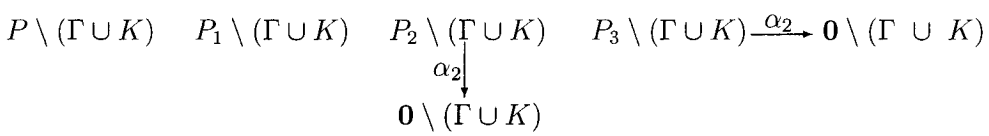

FIG. 5.6 – Sémantique du processus $P \setminus (\Gamma \cup K)$.

5.3 Méthode de preuve par décomposition

Dans cette section, nous démontrons la compositionalité de BNAI par rapport aux principaux opérateurs de SPPA. Tout d'abord, nous avons besoin de la prochaine proposition qui étend un résultat obtenu par Milner [81] pour CCS à notre algèbre de processus SPPA. Cette proposition nous assure que la bisimulation forte est une congruence par rapport aux opérateurs de protocole, de produit parallèle, de restriction et d'observation.

Proposition 5.6 (Congruence). *Soit $L \subseteq Vis$. Si $P_1 \simeq Q_1$ et $P_2 \simeq Q_2$, alors*

1. $P_1 \parallel P_2 \simeq Q_1 \parallel Q_2$;
2. $P_1|P_2 \simeq Q_1|Q_2$;
3. $P_1 \setminus L \simeq Q_1 \setminus L$;
4. $P_1/\mathcal{O}_L \simeq Q_1/\mathcal{O}_L$.

Démonstration. Considérons tout d'abord des relations de bisimulation R_1 et R_2 entre, respectivement, les processus P_1 et Q_1, et les processus P_2 et Q_2.

1. Afin de démontrer cet énoncé, il suffit de voir que la relation

$$\begin{aligned} R = \quad & \{(P_1' \parallel P_2',\ Q_1' \parallel Q_2') \mid (P_1', Q_1') \in R_1 \text{ et } (P_2', Q_2') \in R_2\} \\ & \cup \{(P_1'\#P_2',\ Q_1'\#Q_2') \mid (P_1', Q_1') \in R_1 \text{ et } (P_2', Q_2') \in R_2\} \end{aligned}$$

est une bisimulation entre les processus $P_1 \parallel P_2$ et $Q_1 \parallel Q_2$. D'abord, nous remarquons que $(P_1 \parallel P_2,\ Q_1 \parallel Q_2) \in R$ car $(P_1, Q_1) \in R_1$ et $(P_2, Q_2) \in R_2$. Soit $(P_1' \parallel P_2',\ Q_1' \parallel Q_2') \in R$, d'où $(P_1', Q_1') \in R_1$ et $(P_2', Q_2') \in R_2$. Supposons que $P_1' \parallel P_2' \xrightarrow{\alpha} P$ et α n'est pas une action de marquage. Dans ce cas, nous pouvons supposer que $P_1' \xrightarrow{\alpha} P_1''$ avec $P = P_1'' \parallel P_2'$. Mais puisque $(P_1', Q_1') \in R_1$, il y a une transition $Q_1' \xrightarrow{\alpha} Q_1''$ avec $(P_1'', Q_1'') \in R_1$. D'où $Q_1' \parallel Q_2' \xrightarrow{\alpha} Q_1'' \parallel Q_2'$ avec $(P_1'' \parallel P_2',\ Q_1'' \parallel Q_2') \in R$.

Considérons maintenant le cas où α est une action de marquage. Supposons d'abord que $P_1' \parallel P_2' \xrightarrow{\overline{\delta_{id}^c}(a)} P$, pour certains $c \in C$, $id \in \mathcal{I}$ et $a \in \mathcal{M}$.

Dans ce cas, nous devons avoir $P = P'_1 \# P'_2$ et nous pouvons supposer que $P'_1 \xrightarrow{\overline{c_{id}}(a)} P''_1$ et $P'_2 \xrightarrow{c_{id'}(a)} P''_2$. Puisque $(P'_1, Q'_1) \in R_1$ et $(P'_2, Q'_2) \in R_2$, nous avons $Q'_1 \xrightarrow{\overline{c_{id}}(a)} Q''_1$ et $Q'_2 \xrightarrow{c_{id'}(a)} Q''_2$, avec $(P''_1, Q''_1) \in R_1$ et $(P''_2, Q''_2) \in R_2$. D'où $Q'_1 \parallel Q'_2 \xrightarrow{\delta^c_{id}(a)} Q'_1 \# Q'_2$ avec $(P'_1 \# P'_2,\ Q'_1 \# Q'_2) \in R$. Finalement, supposons que $P'_1 \# P'_2 \xrightarrow{\delta^c_{id}(a)} P$, pour certains $c \in C$, $id \in \mathcal{I}$ et $a \in \mathcal{M}$. Alors nous pouvons supposer que $P'_1 \xrightarrow{\overline{c_{id'}}(a)} P''_1$ et $P'_2 \xrightarrow{c_{id}(a)} P''_2$, avec $P = P''_1 \parallel P''_2$. Comme ci-dessus, nous pouvons conclure qu'il existe une transition $Q'_1 \# Q'_2 \xrightarrow{\delta^c_{id}(a)} Q'_1 \parallel Q'_2$ avec $(P''_1 \parallel P''_2,\ Q''_1 \parallel Q''_2) \in R$. Le cas où nous avons une transition $Q'_1 \parallel Q'_2 \xrightarrow{\alpha} Q$ (ou $Q'_1 \# Q'_2 \xrightarrow{\alpha} Q$) est similaire, ce qui complète la preuve cet énoncé.

2. La démonstration de cet énoncé se déduit facilement de celle de l'énoncé (1).

3. Nous démontrons que la relation

$$R = \{(P'_1 \setminus L,\ Q'_1 \setminus L) \mid (P'_1, Q'_1) \in R_1\}$$

est une bisimulation entre les processus $P_1 \setminus L$ et $Q_1 \setminus L$. Premièrement, nous constatons que $(P_1 \setminus L,\ Q_1 \setminus L) \in R$ car $(P_1, Q_1) \in R_1$. Ensuite, considérons $(P'_1 \setminus L,\ Q'_1 \setminus L) \in R$ et supposons que nous avons une transition $P'_1 \setminus L \xrightarrow{\alpha} P$. Dans ce cas, nous avons $\alpha \notin L$ et $P'_1 \xrightarrow{\alpha} P''_1$, avec $P = P''_1 \setminus L$. Puisque $(P'_1, Q'_1) \in R_1$, il y a une transition $Q'_1 \xrightarrow{\alpha} Q''_1$ avec $(P''_1, Q''_1) \in R_1$. D'où $Q'_1 \setminus L \xrightarrow{\alpha} Q'_1 \setminus L$, avec $(P, Q''_1 \setminus L) \in R$. Le cas où nous avons une transition $Q'_1 \setminus L \xrightarrow{\alpha} Q$ est similaire.

4. Pour cet énoncé, il suffit de démontrer que la relation

$$R = \{(P'_1/\mathcal{O}_L,\ Q'_1/\mathcal{O}_L) \mid (P'_1, Q'_1) \in R_1\}$$

est une bisimulation entre les processus $P_1/\mathcal{O}_L$ et $Q_1/\mathcal{O}_L$. Tout d'abord, nous voyons que $(P_1/\mathcal{O}_L,\ Q_1/\mathcal{O}_L) \in R$. Considérons $(P'_1/\mathcal{O}_L,\ Q'_1/\mathcal{O}_L) \in R$, donc $(P'_1, Q'_1) \in R_1$, et supposons que $P'_1/\mathcal{O}_L \xrightarrow{\alpha} P''_1/\mathcal{O}_L$. Alors, il existe un calcul $P'_1 \xrightarrow{\gamma} P''_1$ avec $\alpha \in \mathcal{O}_L(\gamma)$. Par la Proposition 3.13, nous avons $P'_1 \xrightarrow{\alpha_1} P_1 \ldots \xrightarrow{\alpha_n} P''_1$ avec $\gamma = \alpha_1 \ldots \alpha \ldots \alpha_n$ et $\alpha_i \notin L$ (sauf pour α). Puisque

$(P_1', Q_1') \in R_1$, le processus peut simuler ce calcul, d'où $Q_1' \xrightarrow{\alpha_1} Q_1 \ldots \xrightarrow{\alpha_n} Q_1''$ pour certains processus Q_i tels que $(P_i, Q_i) \in R_1$ et $(P_1'', Q_1'') \in R_1$. Ainsi, nous avons $Q_1' \xrightarrow{\gamma} Q_1''$ avec $(P_1'', Q_1'') \in R_1$. D'où $Q_1'/\mathcal{O}_L \xrightarrow{\alpha} Q_1''/\mathcal{O}_L$ avec $(P_1''/\mathcal{O}_L,\ Q_1''/\mathcal{O}_L) \in R$. Le cas où nous avons une transition $Q_1'/\mathcal{O}_L \xrightarrow{\alpha} Q_1''/\mathcal{O}_L$ est similaire.

□

Dans la prochaine définition, nous considérons une propriété de fermeture sur les ensembles d'actions qui est nécessaire aux résultats de compositionalité présentés plus loin.

Définition 5.7. Soit $L \subseteq Vis$. L'ensemble L est *fermé par rapport à la communication* si

$$\forall_{c\in C}\ \forall_{a\in\mathcal{M}}\ \forall_{id,id'\in\mathcal{I}}\quad \overline{c}_{id}(a) \in L \ \Leftrightarrow\ c_{id'}(a) \in L \ \Leftrightarrow\ \overline{\delta^c_{id}}(a) \in L \ \Leftrightarrow\ \delta^c_{id'}(a) \in L.$$

La prochaine proposition nous permet d'établir la compositionalité de l'opérateur de restriction sur les opérateurs de protocole et de produit parallèle, et une forme faible de compositionalité de l'opérateur d'observation sur ces mêmes opérateurs.

Proposition 5.8 (Compositionalité). *Soit $L \subseteq Vis$ un ensemble fermé par rapport à la communication. Si $P_1 \simeq Q_1$ et $P_2 \simeq Q_2$, alors*

1. $(P_1 \parallel P_2) \setminus L \ \simeq\ (Q_1 \setminus L) \parallel (Q_2 \setminus L)$.
2. $(P_1|P_2) \setminus L \ \simeq\ (Q_1 \setminus L)|(Q_2 \setminus L)$.
3. $(P_1 \parallel P_2)/\mathcal{O}_L \ \simeq_{\text{faible}}\ (Q_1/\mathcal{O}_L) \parallel (Q_2/\mathcal{O}_L)$.
4. $(P_1|P_2)/\mathcal{O}_L \ \simeq_{\text{faible}}\ (Q_1/\mathcal{O}_L)|(Q_2/\mathcal{O}_L)$.

Démonstration. 1. Afin de démontrer cet énoncé, il suffit de montrer que

$$(P_1 \parallel P_2) \setminus L \ \simeq\ (P_1 \setminus L) \parallel (P_2 \setminus L) \tag{5.1}$$

car $(P_1 \setminus L) \parallel (P_2 \setminus L) \simeq (Q_1 \setminus L) \parallel (Q_2 \setminus L)$ par les énoncés (1) et (3) de la Proposition 5.6. Afin d'établir la bisimulation 5.1, nous démontrons que la relation

$$R = \{((P_1' \parallel P_2') \setminus L,\ (P_1' \setminus L) \parallel (P_2' \setminus L)) \mid P_1' \in \mathcal{D}(P_1) \text{ et } P_2' \in \mathcal{D}(P_2)\}$$

est une bisimulation entre les processus $(P_1 \parallel P_2) \setminus L$ et $(P_1 \setminus L) \parallel (P_2 \setminus L)$. Nous commençons par constater que $((P_1 \parallel P_2) \setminus L, (P_1 \setminus L) \parallel (P_2 \setminus L)) \in R$. D'autre part, pour toutes dérivées P_1' et P_2', si nous avons une transition $(P_1' \parallel P_2') \setminus L \xrightarrow{\alpha} P$, et α n'est pas une action de marquage, alors nous pouvons supposer que $P_1' \xrightarrow{\alpha} P_1''$, avec $P = (P_1'' \parallel P_2') \setminus L$. Puisque $\alpha \notin L$, nous avons $P_1' \setminus L \xrightarrow{\alpha} P_1'' \setminus L$, donc

$$(P_1' \setminus L) \parallel (P_2' \setminus L) \xrightarrow{\alpha} (P_1'' \setminus L) \parallel (P_2' \setminus L)$$

avec $((P_1'' \parallel P_2') \setminus L,\ (P_1'' \setminus L) \parallel (P_2' \setminus L)) \in R$.

Supposons maintenant que α est une action de marquage, d'où $(P_1' \parallel P_2') \setminus L \xrightarrow{\overline{\delta_{id}^c}(a)} (P_1' \# P_2') \setminus L$ ou $(P_1' \# P_2') \setminus L \xrightarrow{\delta_{id'}^c(a)} (P_1'' \parallel P_2'') \setminus L$, avec $\overline{\delta_{id}^c}(a), \delta_{id'}^c(a) \notin L$. Dans ce cas, nous pouvons supposer que $P_1' \xrightarrow{\overline{c}_{id}(a)} P_1''$ et $P_2' \xrightarrow{c_{id'}(a)} P_2''$, avec $\overline{c}_{id}(a), c_{id'}(a) \notin L$ car L est fermé par rapport à la communication. Ainsi, nous voyons que $P_1' \setminus L \xrightarrow{\overline{c}_{id}(a)} P_1'' \setminus L$ et $P_2' \setminus L \xrightarrow{c_{id'}(a)} P_2'' \setminus L$. D'où,

$$(P_1' \setminus L) \parallel (P_2' \setminus L) \xrightarrow{\overline{\delta_{id}^c}(a)} (P_1' \setminus L) \# (P_2' \setminus L)$$

ou

$$(P_1' \setminus L) \# (P_2' \setminus L) \xrightarrow{\delta_{id'}^c(a)} (P_1'' \setminus L) \parallel (P_2'' \setminus L).$$

Le cas où nous avons une transition $(P_1' \setminus L) \parallel (P_2' \setminus L) \xrightarrow{\alpha} P$ est similaire.

2. La preuve de cet énoncé se déduit facilement de celle de l'énoncé (1).

3. Afin de démontrer cet énoncé, il suffit de montrer que

$$(P_1 \parallel P_2)/\mathcal{O}_L \simeq (P_1/\mathcal{O}_L) \parallel (P_2/\mathcal{O}_L) \tag{5.2}$$

car $(P_1/\mathcal{O}_L) \parallel (P_2/\mathcal{O}_L) \simeq (Q_1/\mathcal{O}_L) \parallel (Q_2/\mathcal{O}_L)$ par les énoncés (1) et (4) de la Proposition 5.6. Afin d'établir la bisimulation 5.2, nous démontrons que la relation

$$R = \{((P_1' \parallel P_2')/\mathcal{O}_L,\ (P_1'/\mathcal{O}_L) \parallel (P_2'/\mathcal{O}_L)) \mid P_1' \in \mathcal{D}(P_1) \text{ et } P_2' \in \mathcal{D}(P_2)\}$$

est une bisimulation faible entre les processus $(P_1 \parallel P_2)/\mathcal{O}_L$ et $(P_1/\mathcal{O}_L) \parallel (P_2/\mathcal{O}_L)$. Tout d'abord, nous pouvons facilement voir que $((P_1 \parallel P_2)/\mathcal{O}_L,\ (P_1/\mathcal{O}_L) \parallel (P_2/\mathcal{O}_L)) \in R$. Considérons ensuite un couple $((P_1' \parallel P_2')/\mathcal{O}_L,\ (P_1'/\mathcal{O}_L) \parallel (P_2'/\mathcal{O}_L)) \in R$, avec $P_1' \in \mathcal{D}(P_1)$ et $P_2' \in \mathcal{D}(P_2)$, et supposons que nous avons une transition $(P_1' \parallel P_2')/\mathcal{O}_L \xrightarrow{\alpha} P$, où α n'est pas une action de marquage. Dans ce cas, il existe une calcul $(P_1' \parallel P_2') \xrightarrow{\gamma} (P_1'' \parallel P_2'')$ avec $\alpha \in \mathcal{O}(\gamma)$ et $P = (P_1'' \parallel P_2'')/\mathcal{O}_L$. Ainsi, il existe des calculs $P_1' \xrightarrow{\gamma_1} P_1''$ et $P_2' \xrightarrow{\gamma_2} P_2''$ tels que γ_1 et γ_2 sont des sous-suites disjointes de γ dans lesquelles les actions de marquage ont été remplacées par les actions de sorties/entrées correspondantes. De plus, l'action α est présente soit dans γ_1, soit dans γ_2. Nous pouvons donc supposer que $\alpha \in \mathcal{O}(\gamma_1)$ et $\epsilon \in \mathcal{O}(\gamma_2)$ (donc $\tau \in \mathcal{O}(\gamma_2)$), puisque L est fermé par rapport à la communication (d'où γ_1 et γ_2 ne peuvent contenir aucune action de sortie/entrée provenant L). Ainsi $P_1'/\mathcal{O}_L \xrightarrow{\alpha} P_1''/\mathcal{O}_L$ et $P_2'/\mathcal{O}_L \xrightarrow{\tau} P_2''/\mathcal{O}_L$. Nous avons donc $(P_1'/\mathcal{O}_L) \parallel (P_2'/\mathcal{O}_L) \xrightarrow{\alpha\tau} (P_1''/\mathcal{O}_L) \parallel (P_2''/\mathcal{O}_L)$, d'où

$$(P_1'/\mathcal{O}_L) \parallel (P_2'/\mathcal{O}_L) \stackrel{\alpha}{\Longrightarrow} (P_1''/\mathcal{O}_L) \parallel (P_2''/\mathcal{O}_L).$$

Supposons maintenant que nous avons une transition $(P_1' \parallel P_2')/\mathcal{O}_L \xrightarrow{\overline{\delta_{id}^c}(a)} P$ avec $P ::= (P_1''\#P_2')/\mathcal{O}_L$ et $\overline{\delta_{id}^c}(a) \in L$. Comme ci-dessus, nous pouvons supposer que $P_1' \xrightarrow{\gamma_1} P_1''$ et $P_2' \xrightarrow{\gamma_2} P_2''$, avec γ_1 et γ_2 des sous-suites de γ telles

que $\overline{c}_{id}(a) \in \mathcal{O}(\gamma_1)$ et $c_{id'}(a) \in \mathcal{O}(\gamma_2)$. De plus, puisque L est fermé par rapport à la communication, nous avons $\overline{c}_{id}(a), c_{id'}(a) \in L$. D'où $P_1'/\mathcal{O}_L \xrightarrow{\overline{c}_{id}(a)} P_1''/\mathcal{O}_L$ et $P_2'/\mathcal{O}_L \xrightarrow{c_{id'}(a)} P_2''/\mathcal{O}_L$. Donc $(P_1'/\mathcal{O}_L) \parallel (P_2'/\mathcal{O}_L) \xrightarrow{\overline{\delta c}_{id}(a)} (P_1''/\mathcal{O}_L) \parallel (P_2''/\mathcal{O}_L)$. Le cas où nous avons une transition $(P_1'/\mathcal{O}_L)\#(P_2'/\mathcal{O}_L) \xrightarrow{c_{id}(a)} (P_1'' \parallel P_2'')/\mathcal{O}_L$ est similaire au cas précédent. De même, la démonstration est analogue dans le cas où nous avons une transition $(P_1'/\mathcal{O}_L) \parallel (P_2'/\mathcal{O}_L) \xrightarrow{\alpha} P$.

4. La preuve de cet énoncé se déduit facilement de celle de l'énoncé (3).

□

La démonstration du Théorème 5.11 nécessite le prochain corollaire qui établi la compositionalité des opérateurs de restriction et d'observation sur les opérateurs de protocole et de produit parallèle.

Corollaire 5.9. *Soient P et Q des processus et soient L et Γ des sous-ensembles d'action visibles fermés par rapport à la communication. Alors nous avons*

1. $((P \parallel Q) \setminus \Gamma)/\mathcal{O}_L \simeq_{\text{faible}} ((P \setminus \Gamma)/\mathcal{O}_L) \parallel ((Q \setminus \Gamma)/\mathcal{O}_L)$.
2. $((P|Q) \setminus \Gamma)/\mathcal{O}_L \simeq_{\text{faible}} ((P \setminus \Gamma)/\mathcal{O}_L)|((Q \setminus \Gamma)/\mathcal{O}_L)$;

Démonstration. Nous démontrons seulement le premier énoncé car la démonstration du deuxième est similaire. Par l'énoncé (1) de la Proposition 5.8, nous avons une bisimulation

$$(P \parallel Q) \setminus \Gamma \simeq (P \setminus \Gamma) \parallel (Q \setminus \Gamma)$$

et par énoncé (3) de la Proposition 5.8, nous obtenons une bisimulation faible

$$((P \setminus \Gamma) \parallel (Q \setminus \Gamma))/\mathcal{O}_L \simeq_{\text{faible}} ((P \setminus \Gamma)/\mathcal{O}_L) \parallel ((Q \setminus \Gamma)/\mathcal{O}_L).$$

D'où, par l'énoncé (4) de la Proposition 5.6, nous pouvons conclure que

$$((P \parallel Q) \setminus \Gamma)/\mathcal{O}_L \simeq_{\text{faible}} ((P \setminus \Gamma)/\mathcal{O}_L) \parallel ((Q \setminus \Gamma)/\mathcal{O}_L).$$

□

Le lemme suivant nous permet d'établir la commutativité des opérateurs de restriction et d'observation de SPPA par rapport à la bisimulation.

Lemme 5.10. *Soient P et Q des processus et soient L et M des sous-ensembles d'action visibles disjoints. Alors, nous avons*

$$(P/\mathcal{O}_L) \setminus M \simeq (Q/\mathcal{O}_L) \setminus M \quad \textit{si et seulement si} \quad (P \setminus M)/\mathcal{O}_L \simeq (Q \setminus M)/\mathcal{O}_L.$$

Démonstration. Premièrement, si R est une bisimulation entre les processus $(P/\mathcal{O}_L) \setminus M$ et $(Q/\mathcal{O}_L) \setminus M$, alors nous démontrons que la relation

$$R' = \{((P' \setminus M)/\mathcal{O}_L,\ (Q' \setminus M)/\mathcal{O}_L) \mid ((P'/\mathcal{O}_L) \setminus M,\ (Q'/\mathcal{O}_L) \setminus M) \in R\}$$

est une bisimulation entre $(P \setminus M)/\mathcal{O}_L$ et $(Q \setminus M)/\mathcal{O}_L$. Nous constatons d'abord que $((P \setminus M)/\mathcal{O}_L,\ (Q \setminus M)/\mathcal{O}_L) \in R'$ car $((P/\mathcal{O}_L) \setminus M,\ (Q/\mathcal{O}_L) \setminus M) \in R$. Soit $((P' \setminus M)/\mathcal{O}_L,\ (Q' \setminus M)/\mathcal{O}_L) \in R'$ et supposons que nous avons une transition $(P' \setminus M)/\mathcal{O}_L \xrightarrow{\alpha} (P'' \setminus M)/\mathcal{O}_L$. Alors, nous pouvons constater que $\alpha \in L \cup \{\tau\}$, donc $\alpha \notin M$, et nous avons un calcul $P' \setminus M \xrightarrow{\gamma} P'' \setminus M$ avec $\alpha \in \mathcal{O}_L(\gamma)$. Ainsi, nous avons $\gamma \in (Act \setminus M)^*$ (c'est-à-dire la suite d'actions γ ne contient aucune sous-action provenant de M), donc $P' \xrightarrow{\gamma} P''$. Par conséquent, il existe une transition $P'/\mathcal{O}_L \xrightarrow{\alpha} P''/\mathcal{O}_L$ car $\alpha \in \mathcal{O}_L(\gamma)$. D'où $(P'/\mathcal{O}_L) \setminus M \xrightarrow{\alpha} (P''/\mathcal{O}_L) \setminus M$ car $\alpha \notin M$. Puisque R est une bisimulation et $((P'/\mathcal{O}_L) \setminus M,\ (Q'/\mathcal{O}_L) \setminus M) \in R$, il existe une transition $(Q'/\mathcal{O}_L) \setminus M \xrightarrow{\alpha} (Q''/\mathcal{O}_L) \setminus M$ avec $((P''/\mathcal{O}_L) \setminus M,\ (Q''/\mathcal{O}_L) \setminus M) \in R$, d'où $((P'' \setminus M)/\mathcal{O}_L,\ (Q'' \setminus M)/\mathcal{O}_L) \in R'$. Nous avons donc une transition $Q'/\mathcal{O}_L \xrightarrow{\alpha} Q''/\mathcal{O}_L$ car $\alpha \notin M$. Par conséquent, il existe un calcul $Q' \xrightarrow{\gamma'} Q''$ avec $\alpha \in \mathcal{O}_L(\gamma')$, donc $\gamma' \in (Act \setminus M)^*$. Puisque γ' ne contient aucune sous-action provenant de M, nous avons $Q' \setminus M \xrightarrow{\gamma'} Q'' \setminus M$. Ainsi, nous avons une transition $(Q' \setminus M)/\mathcal{O}_L \xrightarrow{\alpha} (Q'' \setminus M)/\mathcal{O}_L$ avec $((P'' \setminus M)/\mathcal{O}_L,\ (Q'' \setminus M)/\mathcal{O}_L) \in R'$. Le cas où nous avons une transition $(Q' \setminus M)/\mathcal{O}_L \xrightarrow{\alpha} (Q'' \setminus M)/\mathcal{O}_L$ est similaire.

Réciproquement, si R est une bisimulation entre les processus $(P \setminus M)/\mathcal{O}_L$ et

$(Q \setminus M)/\mathcal{O}_L$, alors nous pouvons démontrer que la relation

$$R' = \{((P' \setminus M)/\mathcal{O}_L,\ (Q' \setminus M)/\mathcal{O}_L) \mid ((P'/\mathcal{O}_L) \setminus M,\ (Q'/\mathcal{O}_L) \setminus M) \in R\}$$

est une bisimulation entre $(P/\mathcal{O}_L) \setminus M$ et $(Q/\mathcal{O}_L) \setminus M$. La démonstration de cet énoncé est similaire à la démonstration précédente. □

Pour le théorème suivant, nous considérons la propriété BNAI par rapport à des ensembles disjoints K, L et Γ, telle que présentée à la Définition 5.4. De plus, nous supposons que ces trois sous-ensembles d'actions visibles sont fermés par rapport à la communication.

Théorème 5.11 (Compositionalité de BNAI). *Soient P et Q des processus qui satisfont BNAI.*

1. *Le processus $P \setminus M$ satisfait BNAI pour tout $M \subseteq Vis$ fermé par rapport à la communication et tel que $M \cap L = \emptyset$.*
2. *Le processus $P \parallel Q$ satisfait BNAI.*
3. *Le processus $P|Q$ satisfait BNAI.*

Démonstration. 1. Soit $P' \setminus M \in \mathcal{D}(P \setminus M)$, avec $P' \in \mathcal{D}(P)$, et posons $Q ::= P' \setminus \Gamma$ (d'où $P' \setminus (M \cup \Gamma) = Q \setminus M$). Par la Définition 5.4, il suffit de montrer que

$$Q \simeq_{\mathcal{O}_L} Q \setminus K \implies Q \setminus M \simeq_{\mathcal{O}_L} Q \setminus (K \cup M)$$

En effet, nous avons :

$$\begin{aligned} Q \simeq_{\mathcal{O}_L} Q \setminus K &\iff Q/\mathcal{O}_L \simeq (Q \setminus K)/\mathcal{O}_L \\ &\qquad \text{par la Proposition 3.20} \\ &\implies (Q/\mathcal{O}_L) \setminus M \simeq ((Q \setminus K)/\mathcal{O}_L) \setminus M \\ &\qquad \text{par l'énoncé (3) de la Proposition 5.6} \end{aligned}$$

$$\iff (Q \setminus M)/\mathcal{O}_L \simeq (Q \setminus (K \cup M))/\mathcal{O}_L$$

par le Lemme 5.10

$$\iff Q \setminus M \simeq_{\mathcal{O}_L} Q \setminus (K \cup M)$$

par la Proposition 3.20.

2. Soit $P' \parallel Q' \in \mathcal{D}(P \parallel Q)$, avec $P' \in \mathcal{D}(P)$ et $Q' \in \mathcal{D}(Q)$. Puisque les processus P et Q satisfont BNAI, par la Définition 5.4 nous avons

$$P' \setminus \Gamma \simeq_{\mathcal{O}_L} P' \setminus (K \cup \Gamma) \quad \text{et} \quad Q' \setminus \Gamma \simeq_{\mathcal{O}_L} Q' \setminus (K \cup \Gamma)$$

D'où $(P' \setminus \Gamma)/\mathcal{O}_L \simeq (P' \setminus (K \cup \Gamma))/\mathcal{O}_L$ et $(Q' \setminus \Gamma)/\mathcal{O}_L \simeq (Q' \setminus (K \cup \Gamma))/\mathcal{O}_L$ par la Proposition 3.20. Par l'énoncé (1) de la Proposition 5.6, nous avons la bisimulation

$$((P' \setminus \Gamma)/\mathcal{O}_L) \parallel ((Q' \setminus \Gamma)/\mathcal{O}_L) \simeq ((P' \setminus (K \cup \Gamma))/\mathcal{O}_L) \parallel ((Q' \setminus (K \cup \Gamma))/\mathcal{O}_L),$$

et par le Corollaire 5.9, nous voyons que

$$((P' \parallel Q') \setminus \Gamma)/\mathcal{O}_L \simeq_{\text{faible}} ((P' \parallel Q') \setminus (K \cup \Gamma))/\mathcal{O}_L.$$

D'où $(P' \parallel Q') \setminus \Gamma \simeq_{\mathcal{O}_L} (P' \parallel Q') \setminus (K \cup \Gamma)$ par la Proposition 3.21. Finalement, à l'aide de la Définition 5.4, nous pouvons conclure que le processus $P \parallel Q$ satisfait BNAI.

3. La démonstration de cet énoncé est similaire à la démonstration précédente.

□

Notons que ce résultat de compositionalité étend un résultat de Mullins [83] pour la propriété AI dans le contexte de l'algèbre de processus CCS.

5.4 Définissabilité de BNAI dans le μ-calcul

Dans cette section, nous démontrons que la propriété BNAI n'est pas définissable dans le μ-calcul. Bien que BNAI soit définissable par l'entremise de l'équivalence de bisimulation, nous allons constater que c'est une propriété qui ne dépend pas d'une seule trace. Ce résultat nous permet de conclure que BNAI ne correspond pas à une propriété de préservation, ni à une propriété de vivacité. De plus, nous pouvons également constater que BNAI échappe à la famille de *propriétés de sécurité imposables* (*enforceable security policy*) introduite par Schneider [97] (voir Section 2.3.3). Bien que notre résultat soit obtenu pour la propriété BNAI, nous constaterons qu'il est également valide pour d'autres propriétés de non interférence telles BSNNI (Définition 5.1), SNNI (voir Section 2.4.1) et AI (voir Section 2.4.2).

Notons que les résultats obtenus dans ce chapitre sont présentés dans le contexte de l'algèbre de processus SPPA, bien que ceux-ci se transfèrent facilement aux autres algèbres de processus comme CCS et SPA.

5.4.1 μ-calcul modal

Le μ-calcul, initialement proposé par Park [87], est présenté dans cette section comme une extension de la logique modale de Hennessy-Milner. Cette extension s'obtient par l'ajout d'opérateurs de plus grand et de plus petit point fixe dans la syntaxe abstraite de la logique de Hennessy-Milner. Les formules ainsi obtenues permettent, entre autres, de déterminer si un processus peut exécuter des calculs infinis. Cependant, malgré cette expressivité étendue, la satisfaction d'une formule du μ-calcul par un processus est toujours basée, comme c'est le cas pour la logique de Hennessy-Milner, sur les calculs du processus, donc une transition à la fois.

Considérons un ensemble de variables $\{X, Y, Z, \ldots\}$ utilisées pour désigner des ensembles de processus. Les formules du *μ-calcul modal* sont définies selon la syntaxe abstraite suivante :

$$
\begin{array}{rcll}
\varphi & ::= & \mathbf{0} & (\textit{faux}) \\
& | & \mathbf{1} & (\textit{vrai}) \\
& | & X & (\textit{variable de processus}) \\
& | & \neg\varphi & (\textit{négation}) \\
& | & \varphi \vee \varphi & (\textit{disjonction}) \\
& | & \varphi \wedge \varphi & (\textit{conjonction}) \\
& | & \langle L \rangle\ \varphi & (\textit{quantificateur existentiel de transition}) \\
& | & [L]\ \varphi & (\textit{quantificateur universel de transition}) \\
& | & \mu X.F(X) & (\textit{plus petit point fixe}) \\
& | & \nu X.F(X) & (\textit{plus grand point fixe})
\end{array}
$$

pour tout sous-ensemble d'actions $L \subseteq Act$, et où $F(X)$ désigne une application (sur les variables de processus $X, Y, Z, \ldots$) obtenue de la syntaxe du μ-calcul et telle que toute occurrence libre de la variable X dans $F(X)$ est dans la portée d'un nombre pair d'opérateurs de négation $\neg$.

La satisfaction d'une formule close φ par le processus SPPA P, notée par $P \models \varphi$, est définie de façon naturelle pour les formules $\mathbf{0}$, $\mathbf{1}$, $\neg\psi$, $\psi_1 \vee \psi_2$ et $\psi_1 \wedge \psi_2$. Dans le cas où $\varphi ::= \langle L \rangle\ \varphi'$ ou $\varphi ::= [L]\ \varphi'$, nous rappelons que leur satisfaction est définie comme suit :

$$
\begin{array}{ll}
P \models \langle L \rangle\ \psi & \text{si et seulement si } \exists_{Q \in \mathcal{D}_L(P)}\ Q \models \psi; \\
P \models [L]\ \psi & \text{si et seulement si } \forall_{Q \in \mathcal{D}_L(P)}\ Q \models \psi.
\end{array}
$$

où $\mathcal{D}_L(P) = \{Q \in \mathcal{D}(P) \mid \exists_{\alpha \in L}\ P \xrightarrow{\alpha} Q\}$. D'autre part, la satisfaction des opérateurs de point fixe est définie selon le *Théorème de Tarski-Knaster* :

- $P \models \mu X.F(X)$ si et seulement si $P \in R$ pour tout $R \subseteq \mathcal{D}(P)$ tel que $R = \{Q \in \mathcal{D}(P) \mid Q \models F(X)$ où X est interprété comme $R\}$;
- $P \models \nu X.F(X)$ si et seulement si $P \in R$ pour un certain $R \subseteq \mathcal{D}(P)$ tel que $R = \{Q \in \mathcal{D}(P) \mid Q \models F(X)$ où X est interprété comme $R\}$;

où, pour tout $R \subseteq \mathcal{D}(P)$,

$$Q \models R \quad \text{si et seulement si} \quad Q \in R.$$

Janin & Walukiewicz [64] ont introduit une notion de *formules disjonctives* qui possèdent un algorithme linéaire qui permet de vérifier leur satisfaction. De plus, ils démontrent que tout énoncé du μ-calcul est sémantiquement équivalent à une telle formule. Cette notion de formules disjonctives permet donc d'établir une caractérisation du μ-calcul en terme de graphes de transitions finis. D'autre part, Janin & Walukiewicz [65] ont aussi introduit une logique monadique de second ordre pour les graphes de transitions. Ils démontrent que cette logique est invariante par rapport à la bisimulation, tout comme le μ-calcul. En fait, l'invariance du μ-calcul par rapport à la bisimulation est une caractérisation fondamentale de cette logique.

5.4.2 Procédure de décision pour le μ-calcul

Dans cette section, nous présentons une procédure de décision pour le μ-calcul initialement établie par Streett & Emerson [104]. Afin d'obtenir celle-ci, nous devons tout d'abord nous doter d'une description détaillée de la notion de modèle en terme de pré-modèle muni d'une fonction de choix.

Définition 5.12. Le processus P est un *pré-modèle* si, pour toute dérivée $Q \in \mathcal{D}(P)$, nous avons

- $Q \models \neg\varphi$ si et seulement si $Q \not\models \varphi$;
- $Q \models \varphi_1 \vee \varphi_2$ si et seulement si $Q \models \varphi_1$ ou $Q \models \varphi_1$;
- $Q \models \langle L \rangle\ \varphi$ si et seulement si $\exists_{Q' \in \mathcal{D}_L(Q)}\ Q' \models \varphi$;
- $Q \models \mu X.F(X)$ si et seulement si $Q \models F(\mu X.F(X))$.

Une *fonction de choix* pour un pré-modèle P est une fonction qui permet de choisir :

1. pour toute formule $\psi_1 \vee \psi_2$ satisfaite par une dérivée $Q \in \mathcal{D}(P)$, une formule parmi ψ_1 et ψ_2 satisfaite par Q ; et
2. pour toute formule $\langle L \rangle\ \psi$ satisfaite par une dérivée $Q \in \mathcal{D}(P)$, une dérivée $Q' \in \mathcal{D}_L(Q)$ qui satisfait la formule ψ.

Par conséquent, une fonction de choix sur un pré-modèle P détermine une *relation de dérivation* entre les couples (φ, Q), où φ est une formule du μ-calcul et

$Q \in \mathcal{D}(P)$ est une dérivée telle que $Q \models \varphi$. Cette relation de dérivation est définie de façon plus détaillée comme suit :

- $(\psi_1 \vee \psi_2, Q)$ dérive (ψ, Q), où ψ est la formule (ψ_1 ou ψ_2) déterminée par la fonction de choix ;
- $(\psi_1 \wedge \psi_2, Q)$ dérive à la fois (ψ_1, Q) et (ψ_2, Q) ;
- $(\langle L \rangle\ \psi, Q)$ dérive (ψ, Q'), où Q' est la dérivée de Q déterminée par la fonction de choix ;
- $([L]\ \psi, Q)$ dérive (ψ, Q') pour tout Q' tel que $\exists_{\alpha \in L}\ Q \xrightarrow{\alpha} Q'$;
- $(\mu X.F(X), Q)$ dérive $(F(\mu X.F(X)), Q)$;
- $(\nu X.F(X), Q)$ dérive $(F(\nu X.F(X)), Q)$.

Notons que cette relation de dérivation n'est pas réflexive, ni symétrique, ni transitive. Pour les besoins de cette section, nous devons considérer la clôture transitive de cette relation de dérivation. Afin de simplifier notre notation, nous utiliserons la terminologie *dérive* et *suite de dérivation* afin de référer à cette clôture transitive.

À partir des notions définies ci-dessus, il est légitime de se demander si tout modèle correspond à un pré-modèle muni d'une fonction de choix qui n'engendre pas une suite de dérivation qui dérive une même formule $\mu X.F(X)$ une infinité de fois. Cette constatation est presque exacte.

Soient P et Q des processus. Nous disons qu'un énoncé de plus petit point fixe $\mu X.F(X)$ est *régénéré* de P à Q lorsque $(\mu X.F(X), P)$ dérive $(\mu X.F(X), Q)$ et la formule $\mu X.F(X)$ est une sous-formule de toutes les étapes de la dérivation. D'autre part, une fonction de choix est dite *bien fondée* si les relations de régénération pour tous les énoncés de plus petit point fixe, avec leur processus associé, sont bien fondées. Nous disons qu'un pré-modèle est bien fondé s'il est muni d'une fonction de choix bien fondée. Le prochain théorème est un résultat de Streett & Emerson [104].

Théorème 5.13 (Procédure de décision pour le μ-calcul). *Un processus est modèle si et seulement si il est un pré-modèle bien fondé.*

Dans le contexte de cette thèse, nous sommes tout particulièrement intéressés par l'interprétation suivante de ce théorème : si le processus P satisfait une formule

φ du μ-calcul, alors P doit être muni d'une fonction de choix bien fondée qui nous donne la trace du calcul qui nous permet de conclure que $P \models \varphi$. Intuitivement, ce résultat signifie que la satisfaction d'une formule du μ-calcul dépend exclusivement des transitions présentes dans la sémantique du processus.

5.4.3 BNAI n'est pas définissable dans le μ-calcul

Dans cette section, nous démontrons la propriété de sécurité BNAI n'est pas définissable dans le μ-calcul modal, c'est-à-dire qu'il n'existe pas de formule φ dans cette logique telle que P satisfait BNAI si et seulement si $P \models \phi$, pour tout processus P.

Remarque 5.14. Avant de démontrer que la propriété de flots d'information BNAI n'est pas définissable dans le μ-calcul, il est intéressant de remarquer que ce résultat est simple à démontrer lorsque nous nous restreignons à la logique de Hennessy-Milner [1] ou, de façon équivalente, si nous considérons seulement les processus SPPA finis. [2] Dans ce cas, , pour tout entier positif $n \geq 1$, nous pouvons toujours trouver deux processus non bisimilaires mais indistinguables par toute formule de longueur $\leq n$. [3] Cet énoncé est une conséquence du fait qu'une formule de la logique de Hennessy-Milner peut seulement traiter une transition à la fois, et ne peut donc pas distinguer des traces dont la longueur est supérieure à sa longueur. Plus spécifiquement, pour tout $n \geq 1$, il existe des processus distincts P et P' tels que pour toute formule ϕ de la logique modale de Hennessy-Milner, avec $l(\phi) \leq n$, nous avons

$$P \models \phi \iff P' \models \phi.$$

Définition 5.15. Soit P un processus et soient $Q, Q' \in \mathcal{D}(P)$ de dérivées telles que $Q \xrightarrow{\alpha} Q'$, pour une certaine action α. Nous désignons par $P[Q \xrightarrow{\alpha} Q', \alpha']$ le pro-

[1] obtenue en enlevant les opérateurs de point fixe.

[2] processus ayant un nombre fini de dérivées.

[3] La longueur $l(\phi)$ de la formule ϕ est définie par $l(\mathbf{1}) = l(\mathbf{0}) = 1$; $l(\varphi_1 \vee \varphi_2) = l(\varphi_1 \wedge \varphi_2) = l(\varphi_1) + l(\varphi_2) + 1$; $l(\neg\varphi_1) = l(\langle L \rangle\ \varphi_1) = l([L]\ \varphi_1) = l(\varphi_1) + 1$.

cessus obtenu de P en remplaçant la transition $Q \xrightarrow{\alpha} Q'$ par la transition $Q \xrightarrow{\alpha'} Q'$. Autrement, $P[Q \xrightarrow{\alpha} Q', \alpha']$ se comporte comme P.

Lemme 5.16. *Soit φ une formule close du μ-calcul et soit P un processus. Considérons des actions visibles $\alpha, \alpha' \in Vis$ telles que, pour tout sous-ensemble $L \subseteq Act$ présent dans φ, nous avons $\alpha \in L$ si et seulement si $\alpha' \in L$. Alors, pour toute transition $Q \xrightarrow{\alpha} Q'$, avec $Q, Q' \in \mathcal{D}(P)$, nous avons*

$$P \models \varphi \iff P[Q \xrightarrow{\alpha} Q', \alpha'] \models \varphi.$$

Démonstration. Par le Théorème 5.13 et par symétrie, il suffit de démontrer que l'opération de réétiquetage $P[Q \xrightarrow{\alpha} Q', \alpha']$ préserve toute fonction de choix pour P, c'est-à-dire, toute fonction de choix pour P en demeure une pour $P' = P[Q \xrightarrow{\alpha} Q', \alpha']$. De plus, puisque l'unique différence entre ces deux processus est l'action d'une seule transition $Q \xrightarrow{\alpha} Q'$, nous pouvons supposer que $Q = P$.

Soit φ une formule telle que $P \models \varphi$. Nous procédons par induction sur la longueur de φ. Le cas de base $\varphi ::= \mathbf{1}$ est trivial. Supposons que $P \models \psi \iff P' \models \psi$ pour toute formule (close) ψ telle que $l(\psi) < l(\varphi)$. Par le Théorème 5.13, P est un pré-modèle pour φ et il est muni d'une fonction de choix bien fondée. Si φ est une disjonction, alors nous pouvons constater directement de l'hypothèse d'induction que la même fonction de choix fonctionne toujours pour le processus P'. Si $\varphi ::= \langle L \rangle \psi$, alors nous pouvons aussi constater que P' peut conserver la fonction de choix de P car $\alpha \in L$ si et seulement si $\alpha' \in L$. Finalement, nous voyons que la condition que la fonction de choix de P soit bien fondée est préservée lorsque celle-ci est vue comme étant la fonction de choix de P'. Nous pouvons donc conclure que P' est un pré-modèle bien fondé pour la formule φ, d'où $P' \models \varphi$ par le Théorème 5.13. □

Le résultat principal de cette section établit l'impossibilité de définir la propriété de sécurité BNAI dans le μ-calcul modal. Cependant, ce résultat nécessite l'existence d'une infinité d'actions $\alpha, \alpha' \in L$ telles que décrites au Lemme 5.16. De

façon à satisfaire cette condition pour tout énoncé, il suffit d'imposer l'hypothèse plus forte que l'ensemble L est infini. Ainsi, pour ce qui suit nous supposons que $|L| = \infty$.

Théorème 5.17. *La propriété BNAI n'est pas définissable dans le μ-calcul modal.*

Démonstration. Supposons qu'il existe une formule φ du μ-calcul qui permet de définir BNAI. Puisque L est infini, nous pouvons trouver deux actions distinctes $\alpha, \alpha' \in L$ telles que, pour tout sous-ensemble $L_1 \subseteq Act$ présent dans φ, nous avons

$$\alpha \in L_1 \quad \text{si et seulement si} \quad \alpha' \in L_1 \ .$$

Soit $\beta \in K$ et considérons les processus SPPA P et P' dont la sémantique est illustrée à la figure 5.7.

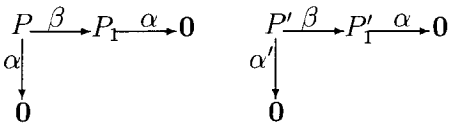

FIG. 5.7 – Sémantique des processus P et P'.

Par le Lemme 5.16 et puisque $P' = P[P \xrightarrow{\alpha} \mathbf{0}, \alpha']$, nous pouvons constater que

$$P \models \varphi \quad \text{si et seulement si} \quad P' \models \varphi \ .$$

Par contre, nous pouvons également constater que le processus P satisfait BNAI, tandis que P' ne satisfait pas BNAI. Ceci constitue une contradiction. Par conséquent, nous pouvons conclure qu'une telle formule φ n'existe pas, ce qui compète la démonstration. □

Il est important de noter que ce résultat a une portée bien plus grande que la simple propriété SNNI. En effet, le Théorème 5.17 est une conséquence de la propriété abstraite de non interférence, et non seulement de son interprétation

BNAI. En particulier, l'énoncé de ce théorème est également valide lorsque nous remplaçons BNAI par les propriétés SNNI, BSNNI et AI. Pour ce convaincre de cette affirmation, il suffit de remarquer que le contre-exemple utilisé dans la preuve du Théorème 5.17 est aussi valide pour ces propriétés. Néanmoins, nous croyons que certaines propriétés de non interférence seraient définissables à l'aide d'un opérateur de second ordre, et par conséquent dans le μ-calcul de second ordre [102].

5.5 Discussion

Nous avons introduit dans ce chapitre une généralisation basée sur la bisimulation de la propriété d'*interférence admissible* (AI), telle que proposée par Mullins [83]. Nous avons également établi une caractérisation algébrique pour cette nouvelle propriété de flots d'information (Définition 5.4), de même que certaines propriétés de compositionalité par rapport aux principaux opérateurs syntaxiques de SPPA (Théorème 5.11). De plus, nous avons démontré que BNAI n'est pas définissable dans le μ-calcul modal (Théorème 5.17).

Des applications de BNAI sont présentées au Chapitre 6, dans lequel nous développons de nouvelles méthodes d'analyse pour les protocoles de sécurité basées sur BNAI. Ces applications généralisent, de façon non triviale, les méthodes basées sur la non interférence proposées par Focardi & Gorrieri [44,48,48,50]. Plus spécifiquement, des propriétés de confidentialité, d'authentification et de vulnérabilité face aux attaques de déni de service sont définies en terme de BNAI et des méthodes de preuve par bisimulation sont déduites pour chacune. Ces propriétés et leur méthodes de preuve sont ensuite illustrées au Chapitre 7 sur des protocoles connus : le protocole *Wide Mouthed Frog* (voir Section 7.2), le protocole d'authentification de Woo-Lam (voir Section 7.3), le protocole d'authentification de Needham-Schröeder (voir Section 7.4), le protocole de branchement TCP/IP (voir Section 7.5), et le protocole de paiement électronique *1KP* (voir Section 7.6).

Un avantage majeur de notre approche, par rapport aux approches basées sur la non interférence, réside dans la capacité d'identifier des attaques inoffensives tôt

dans les phases de conception du protocole. Nous sommes d'avis que cette façon de procéder permet de détecter plus efficacement les failles d'un protocole. D'autre part, dans certains cas, notre approche nous évite d'augmenter notre langage de spécification à l'aide d'un système de déduction parallèle pour traiter les manipulations de messages, dont le cryptage et le décryptage. Ce dernier avantage provient du fait que l'interférence admissible accepte une certaine déclassification de l'information mais uniquement à travers des canaux prévus à cette fin. Ces actions de déclassification s'avèrent ainsi d'excellents outils pour décrire les actions de cryptage d'un protocole cryptographique qui causent inévitablement une interférence, mais celle-ci est acceptable si nous avons confiance à l'algorithme de chiffrement utilisé.

Il serait intéressant de caractériser l'interférence admissible, ou de façon plus générale la non interférence intransitive, dans le contexte du π-calcul. L'obtention de cette caractérisation est principalement motivée par la capacité du π-calcul et de ses diverses extensions de formaliser adéquatement la mobilité, en particulier les applications critiques distribuées que nous retrouvons sur l'Internet. Nous planifions donc une extension de notre méthode d'analyse intransitive des flots d'information qui permettra de traiter des protocoles de sécurité spécifiés dans ces algèbres de processus.

La littérature contient plusieurs modèles basés sur l'algèbre de processus CSP [74,93,100] pour la spécification algébrique et la vérification par *model-checking* des protocoles de sécurité. Cependant, ces modèles nécessitent habituellement la conception d'un intrus suffisamment puissant et spécifique au protocole. Évidement, le choix d'un tel intrus est souvent arbitraire et toute modification au niveau de l'intrus nécessite généralement une nouvelle analyse. Pour la méthode présentée dans cette thèse, nous optons plutôt pour une approche généralisée : l'intrus peut être n'importe quel processus ennemi définissable dans SPPA. Bien qu'elles n'adoptent pas une approche basée sur l'analyse des flots d'information pour caractériser la confidentialité, les méthodes basées sur le spi calcul (présentées à la Section 2.3.5) constituent d'importantes sources d'inspiration pour le développement de notre

méthode, tout particulièrement la façon que ces méthodes surmontent le problème du « plus puissant intrus ».

Pour s'assurer qu'un processus est sécuritaire, nous devons habituellement vérifier qu'une propriété de sécurité est satisfaite même lorsque le processus évolue dans un environnement hostile. Ainsi, le processus doit être en mesure de contrer toute attaque possible provenant d'un intrus, mais la vérification de cette condition est généralement indécidable due au trop grand nombre de stratégies d'attaque possibles. Les méthodes basées sur le spi calcul permettent de contourner ce problème en représentant les propriétés de sécurité à l'aide de formes faibles de l'équivalence de test (voir Section 2.3.5). Intuitivement, deux processus P et Q sont équivalents par test s'il n'existe aucun intrus suffisamment puissant capable de distinguer leurs comportements. Ainsi, cette définition de l'équivalence de test, initialement offerte par Abadi & Gordon [5], souffre d'une quantification universelle qui parcoure tous les environnement hostiles possibles. Boreale, De Nicola & Pugliese [21] proposent une méthode de preuve cohérente et complète qui permet de contourner cette quantification (voir Section 2.3.5). Cette méthode de preuve est essentiellement basée sur une sémantique pour le spi calcul en terme de graphe de transitions étendu et une équivalence de bisimulation faible. Pour des travaux futurs, il serait intéressant d'adapter nos méthodes d'analyse de flots information à ce contexte.

CHAPITRE 6

PROPRIÉTÉS DES PROTOCOLES DE SÉCURITÉ

Dans ce chapitre, nous offrons des interprétations de la propriété BNAI dans le contexte des protocoles de sécurité. Certaines des méthodes que nous introduisons raffinent celles introduites par Focardi, Gorrieri & Martinelli (voir Section 2.4.1). Dans le contexte des propriétés de confidentialité, le concept d'interférence admissible nous procure une meilleure formalisation de l'interférence causée par le cryptage de données. Comme nous l'avons constaté précédemment, ce type d'interférence inoffensive est inévitable lors d'une analyse des flots d'information d'un protocole. Focardi et al. contournent ce problème en considérant une extension syntaxique et sémantique de leur langage de spécification constituée d'un système de règles de déduction parallèle pour traiter séparément les manipulations cryptographiques. L'utilisation d'une approche basée sur l'interférence admissible, combinée à l'algèbre de processus SPPA, nous permet d'effectuer les manipulations cryptographiques requises par le protocole directement sur la sémantique du processus SPPA spécifiant le protocole. L'identification *à priori* de l'interférence qui ne correspond pas à une divulgation significative d'information, donc à un non respect de la confidentialité, est possible grâce à la notion d'actions de déclassification. Nous obtenons ainsi une formalisation de la propriété de confidentialité suivante :

> *Aucun intrus interagissant avec le protocole ne peut différencier, de façon inadmissible, le comportement du protocole et le comportement du protocole dans lequel aucune information confidentielle n'est échangée.*

Focardi, Ghelli & Gorrieri [38,44] ont proposé une méthode dans laquelle les attaques d'authentification sont formalisées en terme d'interférence, plus spécifiquement à l'aide d'une propriété semblable à SNNI (voir Section 2.4.1). Leur méthode consiste essentiellement à s'assurer qu'il n'y a aucun flot d'information entre les participants honnêtes du protocole et tout intrus qui interagit avec le protocole.

Cependant, après leur analyse de non interférence, ils doivent filtrer manuellement certaines interférences qui ne correspondent pas à des attaques d'authentification mais qui sont tout de même détectées. Dans ce contexte, le concept d'actions de déclassification nous permet d'identifier préalablement ces actions qui ne correspondent pas à des attaques d'authentification mais qui causent de l'interférence lors de l'analyse. Par conséquent, une méthode basée sur l'interférence admissible devrait bénéficier d'une efficacité accrue et être plus propice à une automatisation car elle nous permet d'identifier ce type d'interférence avant de procéder à l'analyse, plutôt qu'après. Ainsi, la propriété BNAI nous permet d'établir une formalisation de la propriété d'authentification suivante :

> *Aucun intrus interagissant avec le protocole ne peut interférer de façon inadmissible avec le protocole.*

Dans ce chapitre, nous portons aussi notre attention aux attaques de déni de service (DoS) dans lesquelles un intrus cause un *épuisement de ressources* à une victime (par exemple, un serveur) à travers les étapes d'un protocole de sécurité (habituellement un protocole d'authentification). Dans ce contexte, nous sommes principalement intéressés aux attaques qui nécessitent peu d'effort de la part de l'intrus mais qui causent une grande dépense de ressources pour la victime. Si la victime peut traiter simultanément plusieurs demandes d'exécution du protocole, alors l'intrus peut répéter la même attaque jusqu'au point de causer un épuisement de ressources. Dans ce cas, la victime devra refuser toute nouvelle demande d'exécution du protocole, dont celles provenant de participants honnêtes. Il y a donc déni de service. Afin de détecter les vulnérabilités d'un protocole face à ce type d'attaques de DoS et de DoS distribué, il suffit généralement de vérifier si un seul processus ennemi est capable d'interférer sur les actions coûteuses de la victime en utilisant seulement des actions de coûts minimes. Cette faille isolée qui cause un gaspillage de ressources, lorsque répétée un grand nombre de fois, peut mener à un DoS. Par contre, nous permettons toute interférence provenant d'un intrus qui se comporte convenablement. Cette approche est donc similaire à celle que nous avons adoptée pour notre formalisation de l'authentification. Cette hypothèse

de permettre les comportements honnêtes d'un intrus est souvent omise dans la littérature, mais elle est cruciale afin d'être en mesure de voir l'intrus comme un participant légitime du protocole (c'est-à-dire au même titre que les autres participants). L'interférence admissible nous aide à formaliser cette hypothèse dans un objectif de vérification en permettant à un processus ennemi de causer de l'interférence inoffensive sur le protocole uniquement à partir d'actions prédéterminées appelées attaques admissibles.

Une première formulation des propriétés de confidentialité et d'authentification que nous présentons dans ce chapitre fut initialement proposée dans [84]. Une seconde formulation de la propriété d'authentification fut aussi présentée dans [68]. D'autre part, la propriété de vulnérabilité face aux attaques de DoS que nous introduirons plus bas fut initialement présentée dans [69]. Une seconde formulation plus détaillée est proposée dans [71].

6.1 Confidentialité

L'objectif premier des protocoles de sécurité est la création de canaux sécuritaires permettant des échanges de données confidentielles. Les attaques sur un tel protocole prennent diverses formes, allant des tentatives directes de subtiliser un message confidentiel en entier, aux tentatives plus subtiles d'observer les échanges de données privées afin d'en déduire certaines informations. Dans cette section, nous introduisons une propriété de confidentialité qui permet de détecter tout flot d'information inadmissible pouvant mener à une divulgation non souhaitée de données secrètes. En particulier, notre approche souligne le fait qu'un participant A d'un protocole peut seulement observer les actions qu'il a initiées (contenues dans l'ensemble Act_A), incluant ses actions de sortie et d'entrée effectuées sur les canaux publics. Ainsi, dans le contexte de la confidentialité, nous sommes particulièrement intéressés au critère d'observation $\mathcal{O}_E = \mathcal{O}_{Act_E}$ (défini à la Section 3.2.3), où E désigne un processus ennemi interagissant avec le protocole. Toute modification dans le comportement du protocole observable par un intrus est donc refléter à tra-

vers le critère d'observation $\mathcal{O}_E$. D'où, le concept de $\mathcal{O}_E$-bisimulation nous permet de formaliser le fait qu'un intrus puisse s'apercevoir qu'un protocole échange de l'information confidentielle.

De façon plus précise, notre propriété de confidentialité est obtenue de la propriété BNAI (voir Définition 5.4) en considérant l'ensemble L des actions observables d'un intrus, l'ensemble $K = Act_{\text{secret}}$ des actions qui possèdent un message secret en clair, et l'ensemble Γ des actions de déclassification qui est composé de toute action qui provoque une déclassification admissible de l'information. Les ensembles Act_{secret} et Γ sont formellement définis dans les prochaines sections. En particulier, l'ensemble Act_{secret} nous permettra d'obtenir un critère d'observation $\mathcal{O}_{\text{secret}}$ qui discrimine les actions confidentielles des autres actions. À l'aide de ce critère d'observation, nous pouvons formaliser la dépendance entre les actions contenant un message confidentiel, communément appelées actions de haut-niveau, et les actions observables par l'intrus, communément appelées actions de bas-niveau.

6.1.1 Actions confidentielles

Afin de bien capturer le concept de confidentialité, nous devons étendre la notion d'action SPPA en ajoutant des paramètres à toute action, incluant les appels de fonctions, les extractions, les décryptages et les vérifications de signature. Les paramètres ajoutés sont des messages qui évoquent le contexte de l'action. La syntaxe étendue pour les actions SPPA (obtenue de celle présentée à la Section 3.2.2) est donnée à la Figure 6.1, où $id \in \mathcal{I}$, $f \in \mathcal{F}$ et $a, a', a'' \in \mathcal{M}$. Suivant cette nouvelle syntaxe, la sémantique opérationnelle de SPPA est modifiée de façon naturelle. Par exemple, l'action de cryptage (appel de fonction) $P \xrightarrow{\text{enc}_{id}} P'$ devient l'action $P \xrightarrow{\text{enc}_{id}(k,a)} P'$ si $P ::= \texttt{let}\ x = \text{enc}(k, a)\ \texttt{in}\ P'$; l'action d'extraction $P \xrightarrow{\text{split}_{id}} P'$ devient l'action $P \xrightarrow{\text{split}_{id}(a_1,a_2)} P'$ si $P ::= \texttt{let}\ x = (a_1, a_2)\ \texttt{in}\ P'$; l'action de décryptage $P \xrightarrow{\text{dec}_{id}} P'$ devient l'action $P \xrightarrow{\text{dec}_{id}(k^{-1},\{a\}_k)} P'$ si $P ::= \texttt{case}\ \{a\}_k\ \texttt{of}\ \{a\}_k\ \texttt{in}\ P'$; l'action de vérification de signature $P \xrightarrow{\text{signv}_{id}} P'$ devient l'action $P \xrightarrow{\text{signv}_{id}(k^{-1},a,[a]_k)} P'$ si $P ::= \texttt{case}\ [a]_k\ \texttt{of}\ [a]_k\ \texttt{in}\ P'$; ainsi de suite.

Pour les prochaines définitions, nous considérons un processus ennemi E et un

$$
\begin{array}{rcll}
\alpha & ::= & \overline{c_{id}}(a) & (\textit{sortie}) \\
& | & c_{id}(a) & (\textit{entrée}) \\
& | & f_{id}(a) & (\textit{appel de fonction}) \\
& | & \mathrm{split}_{id}(a, a') & (\textit{extraction}) \\
& | & \mathrm{dec}_{id}(a, a') & (\textit{décryptage}) \\
& | & \mathrm{signv}_{id}(a, a', a'') & (\textit{vérification de signature}) \\
& | & \mathtt{fail}^{\mathtt{f}}_{\mathtt{id}}(\mathtt{a}) & (\textit{échec de fonction}) \\
& | & \mathtt{fail}^{\mathtt{dec}}_{\mathtt{id}}(\mathtt{a}, \mathtt{a}') & (\textit{échec de décryptage}) \\
& | & \mathtt{fail}^{\mathtt{signv}}_{\mathtt{id}}(\mathtt{a}, \mathtt{a}', \mathtt{a}'') & (\textit{échec de vérification de signature}) \\
& | & \mathtt{fail}^{=}_{\mathtt{id}}(\mathtt{a}, \mathtt{a}') & (\textit{échec d'une comparaison}) \\
& | & \delta(a) & (\textit{marquage}) \\
& | & \tau & (\textit{interne})
\end{array}
$$

FIG. 6.1 – Syntaxe étendue des actions SPPA.

ensemble $\mathcal{K}_E \subseteq \mathcal{K}$ qui contient toutes les clés publiques initialement connues par l'intrus.

Définition 6.1. Le message b *contient* le message a (*par rapport à* E), noté par $b \rightsquigarrow a$, si

- $b = a$;
- $b = (b_1, b_2)$, avec $b_1 \rightsquigarrow a$ ou $b_2 \rightsquigarrow a$; ou
- $b = \{b_1\}_k$, avec $b_1 \rightsquigarrow a$ et $k^{-1} \in \mathcal{K}_E$.

Le concept d'un message contenant un autre message s'étend de façon naturelle aux n-tuples $(a_1, \ldots, a_n)$. Nous pouvons constater que tout message de la forme $b = \{a\}_k$ (message chiffré) avec $k^{-1} \notin \mathcal{K}_E$, $b = [a]_k$ (message signé) ou $b = h(a)$ (message haché) ne contient pas le message a, c'est-à-dire $b \not\rightsquigarrow a$. D'autre part, notons que si $b \rightsquigarrow a$ et $a \neq b$, alors on doit avoir $a \not\rightsquigarrow b$.

Définition 6.2. Soit $a \in \mathcal{M}$ un message et soit $\alpha ::= \overline{c_{id}}(b)$, $c_{id}(b)$, $f_{id}(b)$, $\mathrm{split}_{id}(b)$, $\mathrm{dec}_{id}(b)$, $\mathrm{signv}_{id}(b)$ ou $\delta(b)$ une action, ayant $b \in \mathcal{M}$ comme paramètre. L'action α *contient* le message a, noté par $\alpha \rightsquigarrow a$, si

- b est un paramètre de α et $b \rightsquigarrow a$;
- $\alpha ::= f_{id}(b)$ et $f(b) \rightsquigarrow a$, où $f \in \mathcal{F}$ est une fonction ;

- $\alpha ::= f_{id}(-)$ et $a \in \mathrm{im}(f)$, où $f \in \mathcal{F}$ est une fonction génératrice ;
- $\alpha ::= \mathrm{dec}_{id}(k^{-1}, \{b\}_k)$ et $b \rightsquigarrow a$.

L'ensemble $Act(a)$ des actions contenant le message a est donnée par

$$Act(a) \;=\; \{\alpha \in Act \setminus Act_E \mid \alpha \rightsquigarrow a\}.$$

Ainsi, $\alpha \in Act(a)$ dès que l'action α correspond à une manipulation (i.e. réception, envoie, décryptage ou appel de fonction) d'un message qui contient le message a. Ces concepts nous permettrons donc de définir l'ensemble des actions confidentielles. Évidemment, de ce type d'actions nous devons exclure les actions ennemies car, même si elles contiennent a, elles sont considérées comme des actions publiques.

Définition 6.3. Étant donné un ensemble $\mathcal{M}_{\text{secret}} \subseteq \mathcal{M}$ de messages confidentiels, l'*ensemble des actions confidentielles* est défini par

$$Act_{\text{secret}} \;=\; \bigcup_{a \in \mathcal{M}_{\text{secret}}} Act(a).$$

Le contenu exact de l'ensemble $\mathcal{M}_{\text{secret}} \subseteq \mathcal{M}$ des messages confidentiels dépend du protocole à l'étude. De façon générale, il est composé des messages que nous ne souhaitons pas divulguer à un intrus qui interagit le protocole. Pour la suite de ce chapitre, nous fixons un tel ensemble $\mathcal{M}_{\text{secret}}$. Ainsi, de façon à s'assurer que le protocole préserve la confidentialité de tout message provenant de $\mathcal{M}_{\text{secret}}$, nous sommes clairement intéressés à analyser les occurrences des actions provenant de l'ensemble Act_{secret}. Afin de formaliser le concept de confidentialité par rapport à l'ensemble $\mathcal{M}_{\text{secret}}$, nous devons donc considérer le critère d'observation $\mathcal{O}_{\text{secret}} = \mathcal{O}_{Act_{\text{secret}}}$. Par définition, seules les actions correspondant à une manipulation (ne provenant pas de l'intrus) d'un message secret sont observables par $\mathcal{O}_{\text{secret}}$.

6.1.2 Actions de déclassification

Nous rappelons que nous souhaitons obtenir une spécification de la confidentialité qui ne prend pas en considération toute interférence de l'ensemble Act_{secret} sur l'ensemble Act_E qui provient d'une action (prédéterminée) de déclassification. Un tel ensemble Γ d'actions de déclassification contient toutes actions de cryptage (appels de fonction) $\text{enc}_{id}(k, b)$, de même que la plupart des actions de hachage $\text{hash}_{id}(b)$ et des actions de signature $\text{sign}_{id}(k, b)$. Comme nous l'avons constaté précédemment, les actions de cryptage établissent une corrélation admissible entre les messages confidentiels présents dans leurs paramètres et le message chiffré qu'elles retournent dès que le cryptosystème utilisé est suffisamment sécuritaire. Cependant, nous devons être légèrement plus prudent pour les actions de hachage et de signature. Étant donné un message secret $a \in \mathcal{M}_{\text{secret}}$ utilisé lors d'une exécution d'un protocole P, s'il n'y a qu'un petit nombre de messages possibles $a' \in Act_{\text{secret}}$ par lesquels nous pouvons remplacer a dans P, alors le hachage et la signature ne préservent pas la confidentialité de a. Par exemple, si a désigne le résultat d'une certification privée pour un paiement par carte de crédit, $a =$ "*oui*" et $a =$ "*non*", alors le message haché $h(a)$ et le message signé $[a]_k$ (où k^{-1} est une clé publique) ne sont pas sécuritaires. En effet, un intrus pourrait vérifier, pour toutes les valeurs possibles a' pour a (dans ce cas $a' =$ "*oui*" ou $a' =$ "*non*"), si $h(a') = h(a)$, et ainsi « deviner » le contenu du message secret a. De plus, étant donné un message signé $[a]_k$ (qui représente grossièrement le message $\{h(a)\}_k$ où k^{-1} est une clé publique), un processus ennemi pourrait « décrypter » le message $[a]_k$ à l'aide de la clé k^{-1}, obtenir $h(a)$, et ensuite vérifier, pour tout message possible a' pour a, si $h(a') = h(a)$.

Définition 6.4. Soit P un protocole et supposons que a désigne un message secret dans la représentation Alice & Bob de P. Nous disons que a est *suffisamment rare* s'il existe une infinité ou un très grande nombre de messages secrets distincts avec lesquels nous pouvons remplacer a dans P.

Si le message secret a est suffisamment rare, alors les actions de hachage et

les actions de signature correspondantes sont considérées des actions de déclassification. D'autre part, notons qu'une remarque similaire est également valide pour tout message chiffré $\{a\}_k$ où $k \in \mathcal{K}_E$, c'est-à-dire la clé k est supposée initialement connue par un intrus, ce qui inclut les clés publiques.

Définition 6.5. L'ensemble Γ des *actions de déclassification* est composé des actions $\alpha \in Act \setminus Act_E$ qui sastifont l'une des conditions suivantes :

- $\alpha ::= \text{enc}_{id}(k, b)$, où $k \in \mathcal{K}$, $k^{-1} \notin \mathcal{K}_E$ et $b \in \mathcal{M}$;
- $\alpha ::= \text{hash}_{id}(b)$, où tout message secret $a \in \mathcal{M}_{\text{secret}}$ tel que $b \rightsquigarrow a$ est suffisamment rare ; ou
- $\alpha ::= \text{sign}_{id}(k, b)$ où tout message secret $a \in \mathcal{M}_{\text{secret}}$ tel que $b \rightsquigarrow a$ est suffisamment rare ;

avec $id \in \mathcal{I} \setminus \{id_E\}$.

6.1.3 Propriété de confidentialité

À partir des ensembles Act_{secret}, Act_E et Γ définis plus haut, nous pouvous établir la propriété de confidentialité suivante

Définition 6.6 (Confidentialité). Le protocole P est *confidentiel* si, pour tout processus ennemi E,

$$\forall_{Q \in \mathcal{D}(P_E)} \quad Q \setminus \Gamma \simeq_{\mathcal{O}_E} Q \setminus (\Gamma \cup Act_{\text{secret}})$$

Cette propriété peut être vue comme l'interprétation suivante de la propriété de flots d'information BNAI :

$$\forall_{E\text{: processus ennemi}} \quad P_E \text{ satisfait BNAI}$$

avec $K = Act_{\text{secret}}$ et $L = Act_E$ (voir Définition 5.4). (Notons que par la définition de l'ensemble Act_{secret}, les ensembles Act_{secret} et Act_E sont nécessairement disjoints.)

Outre le petit exemple suivant, notre propriété de confidentialité est illustrée à la Section 7.2 à l'aide du protocole *Wide Mouthed Frog* [3], sur lequel une attaque

fut dévoilée il y a quelques années par Abadi [4].

Exemple 6.7. Considérons le protocole $P ::= A \parallel B$, dans lequel les deux participants A et B veulent échanger un message secret a à l'aide d'une clé secrète symétrique k_{AB} (connue seulement des deux participants). Pour cet exemple, nous supposons que tout message est binaire, c'est-à-dire $\mathcal{M} = \{0,1\}^*$. De plus, nous supposons que les comportements des participants A et B sont régis par les participants SPPA $A = (S_A, id_A)$ et $B = (S_B, id_B)$, avec :

$$\begin{aligned} S_A ::= &\ \texttt{let}\ x = \mathrm{enc}(k_{AB}, a)\ \texttt{in}\ \overline{c_1}(x).\mathbf{0} \\ S_B ::= &\ c_1(y_1).\texttt{case}\ y_1\ \texttt{of}\ \{y_2\}_{k_{AB}}\ \texttt{in}\ \texttt{let}\ y_3 = \mathrm{proj}(y_1)\ \texttt{in}\ \overline{c_2}(y_3).\mathbf{0} \end{aligned}$$

où $c_1, c_2 \in C$ sont des canaux publics et $\mathrm{proj} \in \mathcal{F}$ est la fonction de projection du dernier bit (par exemple, $\mathrm{proj}(10010) = 0$). Dans cet exemple, il y a clairement une divulgation inadmissible du message secret a puisque le protocole, plus particulièrement le participant B, révèle de l'information sur le contenu de a (dans ce cas, sa parité), sans pour autant entièrement divulguer a. Une attaque sur la confidentialité du message a peut être perpétrée par tout processus ennemi qui écoute le canal c_2. Par exemple, le processus ennemi

$$E ::= \ c_2(z).\mathbf{0}$$

peut apprendre la parité du message secret a échangé lors du protocole P.

À l'aide de la Définition 6.6, nous pouvons constater que le protocole P, tel que spécifié ci-dessus, ne sastifait pas notre propriété de confidentialité. En effet, soit $A' ::= \overline{c_1}(x).\mathbf{0} \ \in \mathcal{D}(A)$ et considérons le processus dérivé $Q ::= (A' \parallel B \parallel E) \in \mathcal{D}(P_E)$. Alors, nous avons

$$Q \setminus \Gamma \ \not\approx_{\mathcal{O}_E} \ Q \setminus (\Gamma \cup Act_{\text{secret}})$$

car le processus de gauche peut exécuter la calcul

$$Q \setminus \Gamma \xrightarrow{\gamma} \mathbf{0} \parallel \mathbf{0} \parallel \mathbf{0}$$

où

$$\gamma = \overline{\delta^{c_1}_{id_A}}\{a\}_{k_{AB}}\ \delta^{c_1}_{id_B}\{a\}_{k_{AB}}\ \mathrm{dec}_{id_B}(k_{AB}, \{a\}_{k_{AB}})\ \mathrm{proj}_{id_B}(a)\ \overline{\delta^{c_2}_{id_B}}(b)\ \delta^{c_2}_{id_E}(b)$$

avec $b = \mathrm{proj}(a)$. Nous avons donc la transition

$$(Q \setminus \Gamma)/\mathcal{O}_E \xrightarrow{\delta^{c_2}_{id_E}(b)} (\mathbf{0} \parallel \mathbf{0} \parallel \mathbf{0})/\mathcal{O}_E$$

car $\delta^{c_2}_{id_E}(b) \in \mathcal{O}_E(\gamma)$. Or, le processus $Q \setminus (\Gamma \cup Act_{\text{secret}})$ ne peut pas effectuer le calcul correspondant à la suite γ car celle-ci contient une action provenant de l'ensemble Act_{secret}, plus précisément l'action $\mathrm{dec}_{id_B}(k_{AB}, \{a\}_{k_{AB}}) \in Act_{\text{secret}}$. D'où, le processus observé $(Q \setminus (\Gamma \cup Act_{\text{secret}}))/\mathcal{O}_E$ ne peut pas simuler la transition $\xrightarrow{\delta^{c_2}_{id_E}(b)}$, et nous pouvons conclure que $Q \setminus \Gamma \not\approx_{\mathcal{O}_E} Q \setminus (\Gamma \cup Act_{\text{secret}})$.

6.2 Authentification

Une procédure d'authentification est généralement composée d'une phase d'*identification* (un participant revendique une certaine identité) suivie d'une phase de *vérification* (la revendication est vérifiée). Dans un environnement distribué, l'authentification est accomplie par l'entremise de protocoles d'authentification. Dans plusieurs cas, la procédure d'authentification est accompagnée par une distribution de clés de session afin de faciliter les échanges qui suivront l'authentification. Un protocole d'authentification est donc un protocole cryptographique dans lequel un participant A veut authentifier un autre participant B, ou s'authentifier auprès de B. Une attaque perpétrée sur un tel protocole consiste communément en un intrus qui réussit à convaincre B qu'il est A, ou vice versa. Dans plusieurs situations, un participant A doit préalablement initier le protocole avec l'intrus, qui utilisera par la suite les informations obtenues de A afin d'entamer une mascarade envers B.

Dans cette section, nous introduisons une propriété de sécurité qui permet de détecter des failles dans des protocoles d'authentification qui peuvent être exploi-

tées par un intrus afin de perpétrer une attaque. Comme c'était le cas pour la propriété de confidentialité présentée à la section précédente, cette propriété d'authentification est basée sur la propriété BNAI. Nous pourrons ainsi en déduire un théorème de déroulement utile pour des fins de vérification.

Dans le but d'utiliser des méthodes d'analyse de flots d'information pour détecter des attaques sur des protocoles d'authentification, nous devons exprimer toute tentative d'attaque en terme d'interférence sur le protocole. En d'autres termes, nous devons vérifier si un processus ennemi peut causer de l'interférence sur le protocole. Plus spécifiquement, nous souhaitons détecter tout comportement ennemi qui provoque un comportement d'un autre participant qui ne se serait pas produit autrement. De plus, nous restreignons notre procédure de détection seulement aux comportements inadmissibles des processus ennemis et aux comportements critiques des autres participants. La définition exacte de ces deux types de comportements dépend clairement de la spécification du protocole.

6.2.1 Actions critiques et attaques admissibles

Les comportements critiques sont habituellement composés des actions qui correspondent à des états critiques d'un processus, c'est-à-dire les actions qui ne devraient pas se produire lorsque le protocole est attaqué. Ainsi, pour tout participant (honnête) A, nous désignons par $Act_{\text{auth}}(A) \subseteq Act_A$ l'ensemble des actions critiques provenant de A. Nous considérons aussi l'ensemble Act_{auth} des actions critiques du protocole défini par $Act_{\text{auth}} = \bigcup_A Act_{\text{auth}}(A)$. De même, nous considérons le critère d'observation $\mathcal{O}_{\text{auth}} = \mathcal{O}_{Act_{\text{auth}}}$.

La méthode d'analyse de flots d'information présentée dans cette section permet aux processus ennemis de causer de l'interférence inoffensive à travers d'actions prédéterminées, communément appelées *attaques admissibles*. Les actions de déclassification ne jouent donc pas le même rôle dans le contexte d'une propriété d'authentification que dans celui de notre propriété de confidentialité, la situation étant en quelques sortes renversée. Ainsi, nous considérons toute tentative d'attaque

sur le protocole comme de l'interférence, mais nous permettons aux processus ennemis de causer de l'inférence inoffensive par l'entremise de canaux prédéterminés. L'identification, lors de la phase de spécification du protocole, de ces comportements ennemis admissibles nous permet donc d'omettre, lors de l'analyse, toute interférence provenant d'un processus ennemi qui, par exemple, reçoit une invitation pour participer au protocole ou qui initie une exécution honnête du protocole. Cette hypothèse nous permet, entre autres, de voir tout processus ennemi comme un participant légitime du protocole. Notons qu'un tel ensemble d'attaques admissibles, noté par Γ, est un sous-ensemble de l'ensemble Act_E des actions de l'intrus.

6.2.2 Propriété d'authentification

Afin de détecter toute interférence de l'ensemble $Act_E \setminus \Gamma$ sur l'ensemble Act_{auth}, la prochaine propriété d'authentification détermine si tout comportement critique d'un participant honnête (c'est-à-dire A ou B) est indépendant de tout comportement inadmissible provenant d'un processus ennemi.

Définition 6.8 (Authentification). Le protocole P *préserve l'authenticité* si, pour tout processus ennemi E,

$$\forall_{Q \in \mathcal{D}(P_E)} \quad Q \setminus \Gamma \simeq_{\mathcal{O}_{\text{auth}}} Q \setminus Act_E.$$

Notons que cette propriété d'authentification peut être vue comme l'interprétation suivante de la propriété BNAI :

$$\forall_{E\text{: processus ennemi}} \quad P_E \text{ satisfait BNAI}$$

avec $K = Act_E \setminus \Gamma$ et $L = Act_{\text{auth}}$ (selon la notation introduite à la Définition 5.4), où le processus $Q \setminus (Act_E \cup \Gamma)$ correspond au processus $Q \setminus Act_E$ car $\Gamma \subseteq Act_E$. Nous obtenons donc une méthode de preuve basée sur l'équivalence de bisimulation pour notre propriété d'authentification. Celle-ci doit vérifier, pour tout processus ennemi, si tout comportement dans le protocole attaqué qui contient une action critique

demeure un comportement du protocole non attaqué (c'est-à-dire en l'absence de processus ennemis).

Notre propriété d'authentification est illustrée à la Section 7.3 à l'aide du protocole de Woo-Lam [108], et à la Section 7.4 à l'aide du protocole Needham-Schröder [86].

6.3 Déni de service

La contribution principale de cette section est l'introduction d'une méthode d'équivalence-checking pour la validation des protocoles de sécurité contre les attaques de DoS par épuisement de ressources. L'idée fondamentale derrière notre méthode est similaire à celle développée pour notre formalisation de l'authentification et elle consiste à démontrer qu'aucun intrus ne peut utiliser le protocole afin d'interférer avec les actions coûteuses d'un autre participant, dans le but d'épuiser les ressources de celui-ci.

Plus spécifiquement, nous introduisons une propriété de sécurité basée sur BNAI accompagnée d'un algorithme de vérification basé sur la $\mathcal{O}$-bisimulation pour la validation de la robustesse des protocoles de sécurité contre le DoS. Notre méthode utilise évidement l'algèbre de processus SPPA comme langage de spécification. Puisque SPPA nous permet de visualiser explicitement les appels de fonction effectués par un participant comme des actions, nous pouvons leur assigner un coût qui décrit la quantité de ressources requise pour l'exécuter. Il suffit ensuite de vérifier si la spécification du protocole satisfait une propriété de flots d'information appelée *impassibilité* qui n'exige aucune dépendance causale entre les comportements ennemis peu coûteux et les actions coûteuses (c'est-à-dire les actions propices à un épuisement de ressources) provenant des autres participants. En outre, cette propriété de flots d'information détecte si un processus ennemi peut utiliser le protocole afin d'envoyer, avec peu d'effort, un message forgé afin de provoquer une action coûteuse telle une allocation de ressources, le décryptage d'un message ou la vérification d'une signature. Une telle faille pourrait être exploitée par un intrus

de façon à perpétrer une attaque de DoS en gaspillant les ressources précieuses de la victime. Ainsi, l'impassibilité nous assure que le protocole fournit une protection contre les intrus qui ne dépense qu'une petite quantité de ressources.

Notre méthode de validation est basée sur un modèle qui étend SPPA à l'aide d'une notion de coût inspirée d'un modèle proposé par Meadows [78]. Cependant, contrairement à ce modèle, nous ne considérons pas le coût cumulatif des comportements d'un intrus ou des autres participants au cours d'une exécution du protocole. Nous considérons plutôt le coût maximal des actions d'un comportement. D'autre part, la procédure proposée par Meadows pour vérifier si un protocole est vulnérable ou non aux attaques de DoS est essentiellement basée sur une relation de tolérance qui détermine combien de ressources le concepteur du protocole est prêt à dépenser afin d'assurer un certain niveau de sécurité. Dans notre méthode, les capacités d'un intrus sont explicitement formalisées à travers le concept de processus ennemi.

6.3.1 Formalisation des attaques de DoS

Afin de perpétrer une attaque d'épuisement de ressources, un intrus doit habituellement initier plusieurs exécutions du protocole, chacune exploitant la même faille. C'est pour cette raison que nous considérons seulement les attaques où l'intrus est l'instigateur du protocole. D'où, l'interaction d'un processus ennemi E avec le protocole P est spécifiée par le processus $P_E ::= E \parallel B$, où B est la victime (serveur). De plus, nous considérons un ensemble $\Gamma \subseteq Act_E$ d'attaques admissibles, qui contient les actions qui correspondent à des comportements honnêtes de l'intrus (du point de vue du DoS). Ainsi, le processus $P_E \setminus \Gamma$ désigne l'exécution du protocole dans laquelle les comportements honnêtes de l'intrus sont supprimés ; seules les attaques possiblement néfastes pour B sont conservées.

Si nous souhaitons effectuer une analyse du protocole dans un contexte plus général, nous considérons la spécification $P_E ::= E \parallel B \parallel A$, où A est un participant instigateur du protocole. En outre, cette spécification nous permet d'analyser les flots d'information du protocole provoqués par l'intrus et les autres autres partici-

pants honnêtes qui évoluent de manière concurrente.

6.3.2 Fonction de coût

De façon à comparer les dépenses de ressources effectuées par les divers participants à un protocole, nous considérons deux ensembles ordonnés, appelés *ensembles des coûts*, $\langle \mathcal{C}_{\text{cpu}}, < \rangle$ (pour formaliser les ressources CPU) et $\langle \mathcal{C}_{\text{mem}}, < \rangle$ (pour formaliser les ressources mémoire). De plus, nous considérons deux *fonctions de coût*

$$\rho_{\text{cpu}} : Act \rightarrow \mathcal{C}_{\text{cpu}} \quad \text{et} \quad \rho_{\text{mem}} : Act \rightarrow \mathcal{C}_{\text{mem}}$$

où $\rho_{\text{cpu}}(\alpha)$ désigne la quantité de ressources CPU, à savoir le coût CPU, requise pour exécuter l'action α, et $\rho_{\text{mem}}(\alpha)$ désigne la quantité de ressources mémoire, à savoir le coût mémoire, requise pour exécuter l'action α. Notons que la définition de ρ_{mem} peut être généralisée de façon à traiter les suite d'actions dans lesquelles des ressources mémoires sont désallouées, et qui permet ainsi de formaliser une notion de coût cumulatif.

Étant donné un participant B (victime) du protocole, nous considérons sa *capacité CPU* $\text{CPU}_B \in \mathcal{C}_{\text{cpu}}$ qui désigne la capacité du participant du point de vue des ressources CPU :

> *l'exécution simultanée de N actions ayant un coût CPU plus grand que CPU_B risque de causer à B un DoS dû à un épuisement de ressources CPU.*

De façon analogue, nous considérons sa *capacité mémoire* $\text{MEM}_B \in \mathcal{C}_{\text{mem}}$ qui désigne la capacité de B du point de vue des ressources mémoire :

> *l'exécution simultanée de N actions ayant un coût mémoire plus grand que MEM_B risque de causer à B un DoS dû à un épuisement de ressources mémoire.*

Nous considérons également l'ensemble $Act_{\text{coûteux}}$ des actions coûteuses de B, qui

est défini par

$$Act_{\text{coûteux}} = \{\alpha \in Act_B \mid \rho_{\text{cpu}}(\alpha) > \text{CPU}_B \text{ ou } \rho_{\text{mem}}(\alpha) > \text{MEM}_B\}.$$

De plus, nous considérons la capacité CPU de l'intrus, notée par $\text{CPU}_E \in \mathcal{C}_{\text{cpu}}$, qui est définie comme suit :

> *un processus ennemi peut seulement exécuter des actions ayant un coût CPU plus petit ou égal à* CPU_E *;*

de même que sa capacité mémoire, notée par $\text{MEM}_E \in \mathcal{C}_{\text{mem}}$, qui est définie comme suit :

> *un processus ennemi peut seulement exécuter des actions ayant un coût mémoire plus petit ou égal à* MEM_E.

D'où, un intrus qui peut seulement perpétrer des attaques qui nécessitent peu de coût est donc spécifié par un processus ennemi E dans lequel toute transition $E' \xrightarrow{\alpha} E''$, avec $E' \in \mathcal{D}(E)$, est telle que $\rho_{\text{cpu}}(\alpha) \leq \text{CPU}_E$ et $\rho_{\text{mem}}(\alpha) \leq \text{MEM}_E$. Dans ce cas, nous disons que le processus ennemi E *respecte ses capacités*. De toute évidence, les valeurs exactes des capacités de chacun des participants dépendent de plusieurs facteurs techniques qui sont au-delà de la portée de cette thèse. Nous traitons donc ces notions de coût uniquement de manière abstraite.

6.3.3 Impassibilité

Dans cette section, nous introduisons la propriété d'*impassibilité* qui permet de vérifier la robustesse d'un protocole de sécurité contre le DoS, plus spécifiquement de détecter les vulnérabilités du protocole par rapport aux épuisements de ressources CPU et mémoire. Étant donné un serveur B, l'impassibilité détermine si aucun processus ennemi respectant ses capacités ne peut causer de l'interférence inadmissible sur les actions coûteuses $\alpha \in Act_{\text{coûteux}}$.

Définition 6.9 (Impassibilité). Le protocole P est *impassible* si, pour tout pro-

cessus ennemi E respectant ses capacités,

$$\forall_{Q \in \mathcal{D}(P_E)} \quad Q \setminus \Gamma \simeq_{\mathcal{O}_{\text{coûteux}}} Q \setminus Act_E.$$

Notons que cette propriété de DoS peut être vue comme l'interprétation suivante de la propriété BNAI :

$$\forall_{\substack{E : \text{processus ennemi} \\ \text{respectant ses capacités}}} \quad P_E \text{ satisfait BNAI}$$

avec $K = Act_E$ et $L = Act_{\text{coûteux}}$ (selon la notation introduite à la Définition 5.4), où le processus $Q \setminus (Act_E \cup \Gamma)$ correspond au processus $Q \setminus Act_E$ car $\Gamma \subseteq Act_E$. Nous obtenons donc une méthode de preuve cohérente et complète, basée sur l'équivalence de bisimulation, pour notre propriété de DoS. Étant donné un processus ennemi E qui respecte ses capacités, la propriété d'impassibilité est satisfaite dès que la spécification du protocole dans laquelle E tente d'attaquer un participant (B) est bisimilaire, du point de vue des actions coûteuses, à la spécification du protocole dans laquelle les actions de E sont supprimées. En outre, l'impassibilité est satisfaite par tout protocole de sécurité qui utilise une suite de mécanismes d'authentification ordonnés selon des niveaux de sécurité et un coût croissants (pour les deux participants). Lorsqu'un protocole respecte ces critères de conception, un intrus doit être disposé à compléter les premières phases du protocole avant de forcer le système à dépenser des ressources dans des phases avancées du protocole.

Des illustrations de notre propriété d'impassibilité sur des protocoles connus sont offertes au Chapitre 7. À la Section 7.5, nous analysons le *protocole TCP/IP* et nous constatons que l'impassibilité nous permet de redécouvrir une attaque de DoS bien connue sur ce protocole. À la Section 7.6, nous offrons une spécification complète du *protocole de paiement électronique sécuritaire 1KP* [1], que nous analysons ensuite avec notre méthode. Suite à cette analyse, nous démontrons que le

[1] *1KP Secure Electronic Payment Protocol*

protocole 1KP n'est pas impassible et nous dévoilons une vulnérabilité de ce protocole face au DoS par épuisement de ressources. Tout d'abord, nous présentons un protocole d'authentification simple qui ne satisfait pas notre propriété d'impassibilité.

Exemple 6.10. Considérons le protocole suivant :

$$\begin{array}{lccc} \text{Message 1 :} & A & \overset{id_A, n, [id_A, n]_{k_A}}{\longrightarrow} & B \\ \text{Message 2 :} & B & \overset{n}{\longrightarrow} & A\ . \end{array}$$

Dans ce protocole, le participant A initie une procédure d'authentification avec le participant B en lui envoyant le message composé de son identificateur id_A et d'un nonce n fraîchement généré, signé et non signé, où k_A désigne la clé privée de A. Suite à la réception de ce message, B vérifie la signature de A et retourne à ce dernier le nonce n à titre d'accusé de réception. Ce protocole permet donc à un participant A de s'authentifier auprès d'un serveur B.

Nous considérons les participants $A = (S_A, id_A)$ et $B = (S_B, id_B)$, où les agents initiaux S_A et S_B sont définis comme suit :

$$\begin{aligned} S_A ::= \ & \texttt{let}\ x_1 = \text{newNumber}(-)\ \texttt{in}\ \texttt{let}\ x_2 = \text{pair}(id_A, x_1)\ \texttt{in} \\ & \texttt{let}\ x_3 = \text{sign}(k_A, x_2)\ \texttt{in}\ \texttt{let}\ x_4 = \text{pair}(x_2, x_3)\ \texttt{in} \\ & \overline{c_1}(x_4).\ c_2(x_5).[x_5 = x_1]\ \mathbf{0} \end{aligned}$$

$$\begin{aligned} S_B ::= \ & c_1(y_1).\ \texttt{let}\ (y_2, y_3) = y_1\ \texttt{in}\ \texttt{let}\ (y_4, y_5) = y_2\ \texttt{in} \\ & \textstyle\sum_{id \in \mathcal{I}} [y_4 = id]\ \texttt{case}\ y_3\ \texttt{of}\ [y_2]_{k_{id}}\ \texttt{in}\ \overline{c_2}(y_5).\mathbf{0} \end{aligned}$$

où $k_{id_A} = k_A$. Le protocole est alors spécifié par le processus $P ::= A \parallel B$. Cette spécification utilise l'ensemble de fonctions $\mathcal{F} = \{\text{newNumber, pair, sign}\}$. De plus, nous supposons qu'aucune action n'excède la capacité mémoire du serveur B, de même pour la capacité mémoire de l'intrus. Cependant, nous supposons que les actions de vérification de signature (incluant les actions d'échec correspondantes) excèdent la capacité CPU du serveur, d'où

$$\rho_{\text{cpu}}(\text{signv}_{id_B}),\ \rho_{\text{cpu}}(\texttt{fail}^{\text{signv}}_{\texttt{id}_\texttt{B}})\ >\ \text{CPU}_\texttt{B}.$$

De plus, nous supposons que les actions de vérification de signature (dont les actions d'échec) et les actions de signature excèdent la capacité CPU de l'intrus, d'où

$$\rho_{\text{cpu}}(\text{signv}_{id_E}),\ \rho_{\text{cpu}}(\texttt{fail}^{\text{signv}}_{\texttt{id}_\texttt{E}}),\ \rho_{\text{cpu}}(\text{sign}_{\texttt{id}_\texttt{E}}),\ \rho_{\text{cpu}}(\texttt{fail}^{\text{sign}}_{\texttt{id}_\texttt{E}})\ >\ \text{CPU}_\texttt{E}\ .$$

Considérons maintenant le processus ennemi $E = (S_E, id_E)$, avec

$$E ::=\ \texttt{let}\ x_1 = \text{newMessage}(-)\ \texttt{in}\ \texttt{let}\ x_2 = \text{pair}(x_1, x_1)\ \texttt{in}\ \overline{c_1}(x_2).\mathbf{0}\ .$$

Nous pouvons facilement constater que E respecte ses capacités. Ce processus ennemi attaque le protocole en lui envoyant des messages forgés qui forcent B à exécuter des actions coûteuses de vérification de signature (qui vont éventuellement échouer). À l'aide du processus $P_E ::= E \parallel B$, nous pouvons directement constater que le protocole P n'est pas impassible car l'action de marquage $\overline{\delta^{c_1}_{id_E}}((a,a)) \in Act_E$ (pour un message a quelconque) cause de l'interférence sur l'action $\text{signv}_{id_B} \in Act_{\text{coûteux}}$ qui excède la capacité CPU de B.

6.4 Spécification symbolique des propriétés de sécurité

Nous pouvons facilement interpréter les propriétés de sécurités présentées dans ce chapitre dans le contexte de notre modèle symbolique (présenté au chapitre 4) Afin d'obtenir ces spécifications, nous utilisons la notation pour la spécification symbolique des protocoles introduite à la Section 4.3.1.

Nous commençons par voir que notre propriété de confidentialité s'interprète comme suit :

> Le protocole $\langle P,\ \phi_P\rangle$ est *confidentiel* si et seulement si, pour tout processus ennemi $\langle E,\ \phi_E\rangle$,
>
> $$\forall_{\langle Q,\ \psi\rangle \in \mathcal{D}(\langle P_E,\ \phi_{P_E}\rangle)}\quad \langle Q \setminus \Gamma,\ \psi\rangle\ \simeq_{\mathcal{O}_E}\ \langle Q \setminus (\Gamma \cup Act_{\text{secret}}),\ \psi\rangle.$$

Notons que le fait qu'un message secret $a \in \mathcal{M}_{\text{secret}}$ soit suffisamment rare se for-

malise de façon naturelle dans notre modèle symbolique. En effet, un tel message fait généralement parti des connaissances initiales d'un participant A, et peut donc être identifié avant d'amorcer le protocole. Ainsi, nous pouvons remplacer le message secret a par une certaine variable libre x dans la spécification de l'agent initial S_A de A. Dans ce cas, le processus contraint (A, ϕ_A) devra être tel que la formule ϕ_A, qui caractérise les connaissances initiales de A, contient x comme variable libre. Plus spécifiquement, ϕ_A contient une sous-formule ψ qui caractérise le message a, c'est-à-dire ψ est seulement satisfaite par les messages par lesquels nous pouvons remplacer a dans le protocole P. Par exemple, nous pouvons avoir $\psi ::= (x == a)$ si a est unique; $\psi ::= (x == a_1) \wedge \ldots \wedge (x == a_n)$ s'il n'y qu'un nombre fini de possibilités pour a; $\psi ::= \mathcal{N}(x)$ si a peut être n'importe quel nombre. Étant donnée une telle sous-formule ψ, nous pouvons conclure que le message secret a est suffisamment rare si et seulement si la cardinalité de l'ensemble

$$\{a \in \mathcal{M} \mid \models \psi[a/x]\}$$

est suffisamment grande ou infinie.

La spécification des autres propriétés de sécurité est semblable. Ainsi, nous disons que le protocole $\langle P,\ \phi_P \rangle$ *préserve l'authenticité* si et seulement si, pour tout processus ennemi $\langle E,\ \phi_E \rangle$,

$$\forall_{\langle Q,\ \psi \rangle \in \mathcal{D}(\langle P_E,\ \phi_{P_E} \rangle)} \quad \langle Q \setminus \Gamma,\ \psi \rangle \simeq_{\mathcal{O}_{\text{auth}}} \langle Q \setminus Act_E,\ \psi \rangle$$

et le protocole $\langle P,\ \phi_P \rangle$ est *impassible* si et seulement si, pour tout processus ennemi $\langle E,\ \phi_E \rangle$ tel que E respecte ses capacités,

$$\forall_{\langle Q,\ \psi \rangle \in \mathcal{D}(\langle P_E,\ \phi_{P_E} \rangle)} \quad \langle Q \setminus \Gamma,\ \psi \rangle \simeq_{\mathcal{O}_{\text{coûteux}}} \langle Q \setminus Act_E,\ \psi \rangle.$$

6.5 Discussion

La méthode proposée par Focardi & Martinelli [50], permet une formalisation de la confidentialité à l'aide de la propriété de non déductibilité GNDC et de l'algèbre de processus CSPA (voir Section 2.4.1). Leur méthode de validation pour la confidentialité nécessite donc une extension de l'algèbre de processus SPA avec des opérateurs cryptographiques par l'entremise de règles de déduction extérieures à la sémantique opérationnelle de SPA. Nous sommes d'avis que ces manipulations ne sont pas explicitement visibles de la sémantique des processus afin d'éviter de détecter l'interférence admissible causée par le cryptage de données. À l'opposée, notre approche basée sur la propriété de confidentialité (Définition 6.6) et sur l'algèbre de processus SPPA, tend à démontrer qu'il est possible d'utiliser des méthodes de flots d'information sans avoir à étendre la sémantique des algèbres de processus pour éviter l'interférence admissible causée par certaines manipulations cryptographiques. En effet, d'une part SPPA permet une visualisation explicite des manipulations cryptographiques directement dans la sémantique des processus, et, d'autre part, notre propriété de confidentialité, nous permet d'ignorer l'interférence admissible grâce à un choix judicieux d'actions de déclassification.

Le protocole de Needham-Schröder fut préalablement analysé selon une approche de non interférence par Focardi, Ghelli & Gorrieri [44]. Suite à cette analyse, ils découvrent la même faille initialement dévoilée par Lowe et présentée à la Section 7.4. Par ailleurs, notons que leur méthode permet également de redécouvrir la faille dans le protocole de Woo-Lam. D'autre part, Focardi [43] propose une approche différente pour formaliser l'authentification dans laquelle les adresses des entités participant à la procédure d'authentification sont explicites et l'authentification des entités se ramène à une application qui assigne une adresse pour chaque entité. Il suffit ensuite de s'assurer que cette application demeure correcte, même en présence d'un processus ennemi. Cette propriété se vérifie facilement à l'aide d'un outil automatique et peut être utilisée afin de détecter l'attaque sur le protocole de Needham-Schröder. Plus récemment, Broadfoot & Lowe [26] ont proposé

une méthode basée sur CSP qui permet d'étendre certaines techniques connues afin d'analyser les protocoles d'authentification. Leur méthode est illustrée sur le protocole TESLA[2] [89], dans lequel un nombre infini de messages est émit, l'un à la suite de l'autre, par un participant qui possède une quantité illimitée de clés. Dans ce protocole, l'authentification du $n^{\text{ième}}$ message est accomplie à la réception du $(n+1)^{\text{ième}}$ message. Broadfoot & Lowe démontrent que, malgré la nature infinie de ce protocole, il est possible de construire un modèle fini qui formalise adéquatement son comportement.

Les protocoles d'authentification sont développés de façon à fonctionner correctement en présence d'intrus qui peuvent inciter des participants honnêtes à initier un très grand nombre d'exécutions parallèles du protocole. Le nombre d'états nécessaires pour spécifier une seule exécution d'un protocole est typiquement borné et nous pouvons donc espérer obtenir une borne sur le nombre d'exécutions du protocole qui peuvent être utiles à une attaque. Stoller [103] établi une telle borne pour une grande famille de protocoles d'authentification, qui inclut les protocoles de *Needham-Schröder*, de *Otway-Rees* et de *Yahalom*. Dans un même ordre d'idée, Abdulla, Jonsson & Nylén [8] proposent un modèle général pour la formalisation de protocoles d'authentification ayant un nombre infini d'états et un nombre non borné de participants. Ils présentent une méthode de vérification automatique qui est basée sur une analyse d'atteignabilité des états non sécuritaires.

La formalisation de la vulnérabilité face aux DoS présentée dans ce chapitre est basée sur un modèle de coûts qui nous permet d'attribuer des coûts ressources selon les capacités individuelles de chaque participant. Par exemple, si nous souhaitons valider un protocole contre un intrus puissant, nous pouvons supposer que $\rho_{\text{cpu}}(\text{enc}_{id_E}) < \rho_{\text{cpu}}(\text{enc}_{id_B})$, c'est-à-dire le cryptage de données nécessite plus de ressources CPU au serveur B qu'à l'intrus E. D'autre part, nous planifions d'étendre notre modèle de spécification de manière à offrir une meilleure formalisation des attaques de DoS distribuées. En outre, puisqu'une grande partie des attaques de DoS

[2] *Time Efficient Stream Loss-tolerant Authentication Protocol.*

consistent à forcer la victime à allouer plusieurs structures de données dispendieuses (au niveau des ressources mémoire), nous prévoyons étendre notre modèle afin de pouvoir traiter le coût cumulatif des actions effectuées par un même participant.

Toujours à titre de projet, nous prévoyons raffermir les liens entre la méthode d'analyse de vulnérabilité face au DoS présentée dans ce chapitre et le modèle symbolique présenté au Chapitre 4. Cette extension nous semble tout à fait naturelle puisque les attaques de DoS sont essentiellement basées sur la création par un intrus de messages et d'adresses forgés. Or, notre modèle symbolique offre une formalisation adéquate de ces valeurs générées par l'intrus et nous permet d'analyser symboliquement le comportement du protocole lorsque de telles valeurs y sont introduites. Cette approche nous évite donc d'avoir à considérer tous les messages et adresses que l'intrus peut générer. D'autre part, nous avons constaté à la Section 6.4 que nous pouvons facilement obtenir une interprétation de la propriété d'impassibilité dans notre modèle symbolique. Or, nous pouvons aussi exploiter le concept de processus contraint, qui offre un aperçu direct des valeurs symboliques présentes à un certain état du protocole, afin d'établir une nouvelle formalisation de la vulnérabilité face aux attaques de DoS basée sur une analyse d'atteignabilité au niveau du processus contraint correspondant au protocole. Une telle méthode consisterait à s'assurer que pour tout processus contraint $\langle P, \phi \rangle$ correspondant à un état critique du protocole (par exemple, lorsque les ressources d'un participant sont épuisées), la formule ϕ est satisfaite seulement par un petit nombre de messages forgés par l'intrus. Si un protocole satisfait une telle propriété, alors nous pouvons conclure que peu d'attaques perpétrées par un intrus pourront provoquer un DoS. Une autre extension possible à notre modèle de spécification, qui se combine de façon naturelle avec la précédente, consiste à considérer une notion de processus SPPA avec ressources intégrées. Un tel processus serait composé d'un processus SPPA et d'un vecteur qui contient les quantités de ressources disponibles à chacun des participants. Cette approche nous permettra d'obtenir directement à partir de la sémantique du protocole les quantités de ressources (mémoire et CPU) dépensées par un participant, et par conséquent identifier facilement les états critiques

du protocole, c'est-à-dire les états où l'un des participants est propice à un DoS. Ce modèle nous permettra aussi de considérer plus facilement le coût cumulatif des actions effectuées par un participant (principalement du point de vue des ressources mémoire).

CHAPITRE 7

EXEMPLES D'ANALYSES DE PROTOCOLES DE SÉCURITÉ

7.1 Mise en oeuvre des méthodes

Dans cette section, nous présentons les principales étapes de la mise en oeuvre des méthodes de validation présentées au Chapitre 6. Celles-ci seront ensuite illustrées sur des protocoles connus dans les prochaines sections.

Étant donné un protocole, nous devons tout d'abord le spécifier dans SPPA. Nous commençons par identifier les fonctions $f \in \mathcal{F}$ nécessaires à sa modélisation, ainsi que le domaine de chacune de ces fonctions. Ensuite, nous devons identifier et modéliser le comportement de chaque participant à l'aide d'un agent initial SPPA. Étant donné un tel agent S_A, nous devons ensuite lui associer un identificateur id_A et un participant SPPA $A = (S_A, id_A)$. Si nous souhaitons considérer plusieurs participants qui jouent le même rôle que A pour notre analyse, il suffit de produire différents identificateurs $id_{A_1}, \ldots, id_{A_n}$ et différents participants SPPA $A_1 = (S_A, id_{A_1}), \ldots, A_n = (S_A, id_{A_n})$ (qui ont tous le même comportement). Lors de la spécification des participants, nous devons bien identifier les messages qui sont initialement connus de chaque participant (par exemple, les clés publiques et privées) et leur donner des valeurs précises lors de la construction des agents initiaux. Sinon, nous devons recourir au modèle symbolique présenté au Chapitre 4 et spécifier ces valeurs par des variables libres. La Section 4.3.1 contient plus de détails sur la spécification symbolique des protocoles. Nous obtenons alors la spécification du protocole en considérant le processus SPPA qui correspond au produit de tous les participants SPPA (à l'aide de l'opérateur $\|$).

Selon la nature des services offerts par le protocole, nous devons choisir une propriété de sécurité (confidentialité, authentification ou déni de service) selon laquelle nous allons valider le protocole. Dans tous les cas, nous devons définir le contenu des ensembles K, L et Γ.

- Pour la confidentialité, nous devons déterminer l'ensemble $\mathcal{M}_{\text{secret}} \subseteq \mathcal{M}$ des messages confidentiels, à partir duquel nous obtenons l'ensemble $K = Act_{\text{secret}}$ des actions confidentielles. De plus, nous devons définir l'ensemble Γ des actions de déclassification et nous posons $L = Act_E$.
- Pour l'authentification, nous devons déterminer l'ensemble $L = Act_{\text{auth}}$ des actions critiques et l'ensemble Γ des attaques admissibles. De plus, nous posons $K = Act_E$.
- Pour le déni de service, nous devons déterminer les fonctions de coût ρ_{cpu} et ρ_{mem}, de même que les capacités CPU (CPU_A) et les capacités mémoire (MEM_A) de chaque participant (incluant l'intrus). À partir de celles-ci, nous obtenons l'ensemble $L = Act_{\text{coûteux}}$ des actions coûteuses. De plus, nous devons définir l'ensemble Γ des attaques admissibles et nous posons $K = Act_E$.

Ensuite, nous devons modéliser, comme un participant SPPA, un intrus qui peut effectuer un nombre suffisamment grand d'attaques. Nous pouvons toujours définir un processus ennemi générique qui peut « tout faire », mais celui-ci sera nécessairement infini. Si nous avons opté pour l'extension symbolique de SPPA, nous pouvons facilement obtenir une approximation d'un tel processus ennemi qui émet divers messages aléatoires sur les différents canaux publics.

Nous pouvons maintenant procéder à la validation du protocole en utilisant la Définition 6.6, la Définition 6.8 ou la Définition 6.9. L'idée est d'approximer le quantificateur universel qui parcourt l'ensemble des processus ennemis par le processus ennemi obtenu ci-dessus. Si la bisimulation n'est pas satisfaite avec ce processus ennemi, alors nous pouvons dès lors conclure que le protocole ne satisfait pas la propriété de sécurité. Dans ce cas, il est possible de reconstruire l'attaque sur le protocole à partir d'un parcours des états non bisimilaires. Si la bisimulation est satisfaite, alors nous pouvons toujours augmenter le processus ennemi de quelques attaques.

7.2 Protocole Wide Mouthed Frog

Le protocole *Wide Mouthed Frog* est utilisé afin d'établir un canal sécuritaire entre deux participants A et B, sur lequel A souhaite envoyer un message confidentiel a chiffré à l'aide d'une clé de session k. Ce protocole suppose que A et B partagent, respectivement, des clés k_{AS} et k_{BS} avec un troisième participant S (un serveur, par exemple). Le protocole se déroule selon les trois étapes suivantes :

$$\begin{array}{lcccc} \text{Message 1 :} & A & \xrightarrow{id_A, id_B, \{k\}_{k_{AS}}} & S \\ \text{Message 2 :} & S & \xrightarrow{\{id_A, k\}_{k_{BS}}} & B \\ \text{Message 3 :} & A & \xrightarrow{\{a\}_k} & B. \end{array}$$

D'abord, le participant A envoie à S son identificateur (id_A), l'identificateur de B (id_B), ainsi qu'une clé de session fraîche k chiffrée avec la clé permanente k_{AS} (partagée avec S). Ensuite, S décrypte le message reçu $\{k\}_{k_{AS}}$ et envoie à B l'identificateur de A avec la clé k chiffrés ensemble à l'aide de la clé partagée k_{BS}. Finalement, A envoie, directement à B, son message secret a chiffré avec la clé k. Le participant B peut maintenant décrypter le message $\{id_A, k\}_{k_{BS}}$ afin d'obtenir la clé k, puis décrypter le message $\{a\}_k$ afin d'obtenir le message secret a.

Une attaque bien connue sur le protocole *Wide Mouthed Frog*, initialement découverte par Abadi [4], peut être perpétrée par un processus ennemi E comme suit. D'abord, E intercepte le *Message 1*, puis substitue l'identificateur de B avec son propre identificateur et envoie ce nouveau message à S. Suite à la réception de ce message, le participant S est persuadé que A veut donner la clé de session k à E, et non à B. Ainsi, S envoie le message $\{id_A, k\}_{k_{ES}}$ à E qui peut le décrypter et par conséquent obtenir la clé k. Il suffit maintenant à l'intrus d'intercepter et de décrypter le *Message 3* afin d'acquérir le message confidentiel a. Cette attaque est présentée de façon plus détaillée à la Section 7.2.2, dans laquelle nous en offrons une spécification complète dans l'algèbre de processus SPPA. Tout d'abord, nous présentons dans la prochaine section une spécification des participants A, B et S.

7.2.1 Spécification du protocole Wide Mouthed Frog

Considérerons les participants SPPA $A = (S_A, id_A)$, $B = (S_B, id_B)$ et $S = (S_S, id_S)$, où les agents initiaux S_A, S_B et S_S sont donnés par :

$$\begin{aligned} S_A ::= \ & \texttt{let } x_1 = \text{newKey}(-) \texttt{ in let } x_2 = \text{enc}(k_{AS}, x_1) \texttt{ in} \\ & \texttt{let } x_3 = \text{pair}(id_A, id_B, x_2) \texttt{ in } \overline{c_1}(x_3). \\ & \texttt{let } x_4 = \text{enc}(x_1, a) \texttt{ in } \overline{c_3}(x_4).\mathbf{0} \end{aligned}$$

$$\begin{aligned} S_B ::= \ & c_2(y_1).\texttt{case } y_1 \texttt{ of } \{y_2\}_{k_{BS}} \texttt{ in let } (y_3, y_4) = y_2 \texttt{ in} \\ & c_3(y_5).\texttt{case } y_5 \texttt{ of } \{y_6\}_{y_4} \texttt{ in } \mathbf{0} \end{aligned}$$

$$\begin{aligned} S_S ::= \ & c_1(z_1).\texttt{let } (z_2, z_3, z_4) = z_1 \texttt{ in} \\ & \textstyle\sum_{id \in \mathcal{I}}[z_2 = id] \texttt{ case } z_4 \texttt{ of } \{z_5\}_{k_{idS}} \texttt{ in} \\ & \textstyle\sum_{id' \in \mathcal{I}}[z_3 = id'] \texttt{ let } z_6 = \text{pair}(z_2, z_5) \texttt{ in} \\ & \texttt{let } z_7 = \text{enc}(k_{id'S}, z_6) \texttt{ in } \overline{c_2}(z_7).S \end{aligned}$$

où $c_1, c_2, c_3 \in C$ sont les canaux publics sur lesquels les messages 1, 2 et 3 sont respectivement échangés. (Nous utilisons la notation $k_{id_AS} = k_{AS}$, etc.) Pour cet exemple, il suffit de considérer l'ensemble de fonctions $\mathcal{F} = \{\text{enc}, \text{pair}\}$. Le protocole *Wide Mouthed Frog* est donc spécifié par le processus SPPA :

$$P ::= A \parallel B \parallel C$$

avec $\mathcal{M}_{\text{secret}} = \{k, a\}$.

7.2.2 Spécification de l'attaque sur le protocole Wide Mouthed Frog

Une alternative intéressante au quantificateur universel « *pour tout processus ennemi E* » présent dans la définition de notre propriété de confidentialité, consiste à obtenir un plus puissant intrus. Une telle approche fut présentée dans un papier précèdent [68]. Par contre, dans cet exemple, un tel plus puissant intrus n'est pas nécessaire puisque que nous souhaitons démontrer que le protocole n'est pas confidentiel et non la réciproque. Ainsi, pour atteindre cet objectif il suffit de trouver

un certain processus ennemi E pour lequel la Définition 6.6 n'est pas valide. Pour ce besoin, nous spécifions avec SPPA le processus ennemi utilisé par Focardi & Martinelli [50] et qui correspond à l'attaque mentionnée plus haut. Ce processus ennemi correspond au participant SPPA $E = (S_E, id_E)$ avec

$$\begin{aligned} S_E ::= \; & c_1(w_1).\texttt{let}\ (w_2, w_3, w_4) = w_1\ \texttt{in}\ \texttt{let}\ w_5 = \mathrm{pair}(w_2, id_E, w_4)\ \texttt{in} \\ & \overline{c_1}(w_5).\ c_2(w_6).\texttt{case}\ w_6\ \texttt{of}\ \{w_7\}_{k_{ES}}\ \texttt{in}\ \texttt{let}\ (w_8, w_9) = w_7\ \texttt{in} \\ & c_3(w_{10}).\texttt{case}\ w_{10}\ \texttt{of}\ \{w_{11}\}_{w_9}\ \texttt{in}\ \mathbf{0} \end{aligned}$$

À partir de ce processus ennemi, nous pouvons vérifier que le protocole *Wide Mouthed Frog*, spécifié par le processus P, n'est pas confidentiel. En effet, il suffit de considérer la dérivée $Q ::= A' \parallel B \parallel S' \parallel E' \in \mathcal{D}(P_E)$ où les participants $A' = (S_1, id_A) \in \mathcal{D}(A)$, $S' = (S_2, id_S) \in \mathcal{D}(S)$ et $E' = (S_3, id_E) \in \mathcal{D}(E)$ sont tels que

$$S_1 ::= \; \texttt{let}\ x_4 = \mathrm{enc}(k, a)\ \texttt{in}\ \overline{c_3}(x_4).\mathbf{0}$$

$$S_2 ::= \; \overline{c_2}(\{(id_a, k)\}_{k_{ES}}).S$$

$$\begin{aligned} S_3 ::= \; & c_2(w_6).\texttt{case}\ w_6\ \texttt{of}\ \{w_7\}_{k_{ES}}\ \texttt{in}\ \texttt{let}\ (w_8, w_9) = w_7\ \texttt{in} \\ & c_3(w_{10}).\texttt{case}\ w_{10}\ \texttt{of}\ \{w_{11}\}_{w_9}\ \texttt{in}\ \mathbf{0} \end{aligned}$$

pour une certaine clé $k \in \mathrm{im}(\mathrm{newKey})$ générée par le participant A. À partir de ce processus Q, nous pouvons constater que

$$Q \xrightarrow{\overline{\delta^{c_2}_{id_S}}(\{(id_a,k)\}_{k_{ES}})} Q' \xrightarrow{\delta^{c_2}_{id_E}(\{(id_a,k)\}_{k_{ES}})} Q''$$

avec $\overline{\delta^{c_2}_{id_S}}(\{(id_a, k)\}_{k_{ES}}) \in Act_{\text{secret}}$ (car $k_{ES} \in \mathcal{K}_E$) et $\delta^{c_2}_{id_E}(\{(id_a, k)\}_{k_{ES}}) \in Act_E$. Puisque $\mathcal{O}_E(\overline{\delta^{c_2}_{id_S}}(\{(id_a, k)\}_{k_{ES}})\delta^{c_2}_{id_E}(\{(id_a, k)\}_{k_{ES}})) = \delta^{c_2}_{id_E}(\{(id_a, k)\}_{k_{ES}})$, nous avons la transition

$$(Q \setminus \Gamma)/\mathcal{O}_E \xrightarrow{\delta^{c_2}_{id_E}(\{(id_a,k)\}_{k_{ES}})} (Q'' \setminus \Gamma)/\mathcal{O}_E$$

tandis que le processus $(Q \setminus (\Gamma \cup Act_{\text{secret}}))/\mathcal{O}_E$ ne peut simuler cette transition. Ainsi, nous avons $(Q \setminus \Gamma)/\mathcal{O}_E \not\approx (Q \setminus (\Gamma \cup Act_{\text{secret}}))/\mathcal{O}_E$, d'où $Q \setminus \Gamma \not\approx_{\mathcal{O}_E} Q \setminus (\Gamma \cup$

Act_{secret}). Nous pouvons donc conclure que le protocole P n'est pas confidentiel.

7.3 Protocole de Woo-Lam

Dans cette section, nous illustrons notre propriété d'authentification à l'aide du *protocole d'authentification unilatérale de Woo-Lam* [108]. Cette application spécifique nous permet du même coup d'illustrer comment le concept d'interférence admissible nous permet d'accomplir une identification, à la phase de spécification, des interférences possibles causées par des processus ennemis et qui ne correspondent pas à des attaques réussites. Notons que ces interférences admissibles, appelées plus haut attaques admissibles, sont détectées lors de l'analyse de flots d'information effectuée par Focardi & al. [38] ; mais ça ne devrait pas être le cas puisqu'elles ne correspondent pas à de vraies attaques.

Le protocole de Woo-Lam est initié par un participant A qui souhaite s'authentifier auprès d'un autre participant B. Ce protocole suppose que les participants A et B partagent chacun une clé symétrique permanente, désignées respectivement par k_{AS} et k_{BS}, avec un troisième participant S en qui ils ont confiance (par exemple, un serveur). Le protocole se déroule selon les étapes suivantes :

$$
\begin{array}{lcccc}
\text{Message 1 :} & A & \xrightarrow{id_A} & B \\
\text{Message 2 :} & B & \xrightarrow{n_B} & A \\
\text{Message 3 :} & A & \xrightarrow{\{n_B\}_{k_{AS}}} & B \\
\text{Message 4 :} & B & \xrightarrow{\{id_A,\{n_B\}_{k_{AS}}\}_{k_{BS}}} & S \\
\text{Message 5 :} & S & \xrightarrow{\{n_B\}_{k_{BS}}} & B.
\end{array}
$$

D'abord, le participant A initie le protocole avec B en lui envoyant son identificateur id_A. Ensuite, le participant B répond en lui retournant un nonce n_B fraîchement généré. Suite à la réception de ce dernier message, A retourne à B le nonce n_B chiffré avec la clé k_{AS}. Le participant B peut maintenant débuter la procédure qui lui permettra d'authentifier A avec l'aide de S, en lui envoyant l'identificateur de A et le dernier message obtenu de A, chiffrés ensemble avec la clé k_{BS}. Le participant S décrypte le message reçu de B à l'aide de la clé k_{BS}, puis décrypte le message

$\{n_B\}_{k_{AS}}$ l'aide de la clé k_{AS}. Finalement, S envoie n_B chiffré avec la clé k_{BS} à B. Une fois ce message reçu et décrypté par B, il lui suffit de s'assurer que la valeur ainsi obtenue correspond bien au nonce n_B qu'il a initialement envoyé à A. Si c'est le cas, B peut conclure qu'il a authentifié A, ou presque... En effet, cette authentification n'est pas toujours exacte puisque qu'une faille dans le protocole de Woo-Lam fut découverte par Abadi (et rapportée par Woo & Lam [109]). Dans cette attaque de type « *man-in-the-middle* », le participant A initie le protocole avec un processus ennemi E qui utilise l'information obtenue de A afin d'initier le protocole avec un autre participant B et se faire passer pour A. Cette attaque est présentée de façon détaillée à la Section 7.3.2. Tout d'abord, nous offrons une spécification complète du protocole dans l'algèbre de processus SPPA.

7.3.1 Spécification du protocole de Woo-Lam

Afin de spécifier le protocole d'authentification de Woo-Lam dans SPPA, nous considérons les participants SPPA $A = (S_A, id_A)$, $B = (S_B, id_B)$ et $S = (S_S, id_S)$ dont les agents initiaux sont définis comme suit :

$$S_A ::= \sum_{id \in \mathcal{I}} \overline{c_{1id_B}}(id_A).\ c_{2id_A}(x_1).\ \texttt{let}\ x_2 = \text{enc}(k_{AS}, x_1)\ \texttt{in}\ \overline{c_{3id_B}}(x_2).\mathbf{0}$$

$$\begin{aligned} S_B ::= \ & c_{1B}(y_1).\ \texttt{let}\ y_2 = \text{newNumber}(-)\ \texttt{in}\ \overline{c_{2y_1}}(y_2).\ c_{3B}(y_3). \\ & \texttt{let}\ y_4 = \text{pair}(y_1, y_3)\ \texttt{in let}\ y_5 = \text{enc}(k_{BS}, y_4)\ \texttt{in} \\ & \overline{c_4}(y_5).\ c_{5id_B}(y_6).\ \texttt{case}\ y_6\ \texttt{of}\ \{y_7\}_{k_{BS}}\ \texttt{in} \\ & [y_7 = n]\ \texttt{let}\ y_8 = \text{auth}(y_1)\ \texttt{in}\ \mathbf{0} \end{aligned}$$

$$\begin{aligned} S_S ::= \ & c_4(z_1).\ \sum_{id \in \mathcal{I}} \texttt{case}\ z_1\ \texttt{of}\ \{z_2\}_{k_{idS}}\ \texttt{in let}\ (y_3, y_4) = z_2\ \texttt{in} \\ & \texttt{case}\ z_4\ \texttt{of}\ \{z_5\}_{k_{Y_3S}}\ \texttt{in let}\ z_6 = \text{enc}(k_{idS}, z_5)\ \texttt{in}\ \overline{c_{5id}}(z_6).S \end{aligned}$$

où c_{jid} désigne canal public utilisé pour envoyer le $j^{\text{ième}}$ message du protocole destiné au participant associé à l'identificateur id (mais qui peut être intercepté par un intrus). Nous pouvons désigner le $4^{\text{ième}}$ canal simplement par c_4 puisque nous supposons qu'il est toujours destiné au serveur S. D'autre part, nous utilisons

la notation $\sum_{id \in \mathcal{I}}$ dans la spécification de A et de S afin de représenter un choix non déterministe de l'identificateur d'un autre participant. Notons également que notre spécification du participant B est telle que dès qu'il termine avec succès une exécution du protocole (c'est-à-dire lorsqu'il croit avoir authentifié le participant A), il exécute l'appel de fonction $\text{auth}(id_A)$ qui enregistre l'identificateur de A (dans la mémoire locale de B) comme étant authentifié. D'où, nous posons $\text{auth}(id) = id$, avec $\phi_{\text{auth}} ::= \mathcal{I}(x)$.

À l'aide de ces spécifications, le protocole de Woo-Lam est défini par le processus SPPA suivant :

$$P ::= A \parallel B \parallel S$$

avec $C = \bigcup_{id \in \mathcal{I}} \{c_{1id}, c_{2id}, c_{3id}, c_4, c_{5id}\}$. Pour des fins de vérification, nous supposons que les seules actions critiques (du point de vue de l'authentification) correspondent aux appels de fonction $\text{auth}_{id_B} \in Act_B$. D'où $Act_{\text{auth}} = \{\text{auth}_{id_B}\}$. Nous ne définissons pas immédiatement les actions qui composent l'ensemble Γ puisque nous interprétons ces attaques admissibles comme de l'interférence admissible provenant d'un processus ennemi, d'où $\Gamma \subseteq Act_E$.

7.3.2 Spécification de l'attaque sur le protocole de Woo-Lam

Dans cette section, nous utilisons la propriété de flots d'information BNAI afin de découvrir la faille dévoilée par Abadi [109] dans le protocole de Woo-Lam. Notre spécification de cette attaque est inspirée de la spécification offerte par Durante, Focardi & Gorrieri [38]. Plus précisément, nous démontrons que le protocole d'authentification unilatérale de Woo-Lam ne préserve pas l'authenticité. Afin d'établir ce fait, nous considérons le prochain processus ennemi qui effectue l'attaque « *man-in-the-middle* » d'Abadi. Ce processus ennemi est donné par le participant SPPA $E = (S_E, id_E)$, où l'agent initial S_E est défini par :

$$S_E ::= c_{1id_E}(w_1).\ \overline{c_{1id_B}}(w_1).\ c_{2w_1}(w_2).\ \overline{c_{2w_1}}(w_2).\ c_{3id_E}(w_3).\ \overline{c_{3id_B}}(w_3).\mathbf{0}\ .$$

Par ailleurs, le protocole attaqué est donné par le processus SPPA :

$$P_E ::= A \parallel B \parallel S \parallel E$$

dans lequel le participant A peut initier le protocole avec B ou avec E. En particulier, quand A tente de s'authentifier auprès de E, ce dernier réutilise les données obtenues de A afin de lui subtiliser son identité et, par conséquent, s'authentifier en tant que A auprès de B. À la fin de cette exécution du protocole, B est persuadé que E est A.

7.3.3 Attaques admissibles

Afin d'identifier correctement les actions qui correspondent à des attaques admissibles dans notre spécification du protocole de Woo-Lam, nous devons considérer toute action ennemie correspondant à un comportement honnête. Par exemple, si un processus ennemi initie le protocole en utilisant son propre identificateur ou s'il reçoit une demande d'authentification de la part d'un autre participant, alors nous devons identifier les actions correspondant à ces situations comme étant des attaques admissibles, c'est-à-dire des actions ennemies que nous ne souhaitons prendre en considération lors de notre analyse des flots d'information du protocole. À partir du processus ennemi E spécifié plus haut, nous pouvons constater que toute action de marquage d'entrée $\delta^c_{id_E}(a)$ devrait être considérée comme une attaque admissible. De plus, toute action de marquage de sortie $\overline{\delta^c_{id_E}}(a)$ telle que le message a contient l'identificateur id_E (et non celui de A ou un identificateur forgé) devrait aussi être considérée comme une attaque admissible. En fait, puisque nous ne souhaitons pas détecter l'interférence provenant d'une exécution honnête du protocole par l'intrus, nous considérons comme attaques admissibles toutes les marqueurs de sortie suivants (pour tout $id \in \mathcal{I}$) :

1. $\overline{\delta^{c_{1id}}_{id_E}}(a)$ tel que $\models a == id_E$;
2. $\overline{\delta^{c_{2id}}_{id_E}}(a)$ tel que $\models \mathcal{N}(a)$ (c'est-à-dire a doit être un nonce) ;
3. $\overline{\delta^{c_{3id}}_{id_E}}(a)$ tel que $\models \exists_x \, (a == \{x\}_{k_{ES}} \wedge \mathcal{N}(x))$;

4. $\overline{\delta^{c_4}_{id_E}}(a)$ tel que $\models \exists_{x,y,z}\ (a == \{(z,\{x\}_y)\}_{k_{ES}} \wedge \mathcal{N}(x) \wedge \mathcal{K}(y) \wedge \mathcal{I}(z))$.

Notons qu'aucun marqueur de sortie de la forme $\overline{\delta^{c_{5id}}_{id_E}}(a)$ est considéré admissible car le serveur S est le seul participant autorisé à envoyer un messages sur les canaux publics c_{5id}.

Les actions correspondant à des appels de fonction tels le cryptage enc_{id_E}, le décryptage dec_{id_E} et la génération de nombres aléatoires newNumber_{id_E}, sont également considérées comme étant des attaques admissibles car elles sont nécessaires à tout processus ennemi qui souhaite participer honnêtement au protocole. Ainsi, nous pouvons supposer que l'ensemble Γ des attaques admissibles est composé de toutes les actions de l'ensemble Act_E (les actions ennemies), à l'exception des actions de marquage de sortie qui ne sont pas énumérées plus haut.

7.3.4 Analyse de flots d'information du protocole de Woo-Lam

À l'aide de la Définition 6.8, nous pouvons constater que le protocole P ne préserve pas l'authenticité car

$$Q \setminus \Gamma \not\approx_{\mathcal{O}_{\text{auth}}} Q \setminus Act_E$$

pour un certain $Q \in \mathcal{D}(P_E)$. En effet, un tel processus Q est donné par la dérivée du processus P_E qui peut effectué le calcul de la suite d'actions suivantes :

$$\begin{aligned}\gamma = \ & \overline{\delta^{c_{3B}}_{id_E}}(\{n_B\}_{k_{AS}})\ \delta^{c_{3B}}_{id_B}(\{n_B\}_{k_{AS}})\ \text{pair}_{id_B}\ \text{enc}_{id_B}\ \overline{\delta^{c_4}_{id_B}}(\{id_A,\{n_B\}_{k_{AS}}\}_{k_{BS}}) \\ & \delta^{c_4}_{id_S}(\{id_A,\{n_B\}_{k_{AS}}\}_{k_{BS}})\ \text{split}_{id_S}\ \text{dec}_{id_S}\ \text{enc}_{id_S}\ \overline{\delta^{c_{5B}}_{id_S}}(\{n_B\}_{k_{BS}}) \\ & \delta^{c_{5B}}_{id_B}(\{n_B\}_{k_{BS}})\ \text{dec}_{id_B}\ \text{auth}_{id_B}.\end{aligned}$$

Nous pouvons constater que $\overline{\delta^{c_{3B}}_{id_E}}(\{n_B\}_{k_{AS}}) \in Act_E \setminus \Gamma$ et $\text{auth}_{id_B} \in Act_{\text{auth}}$, donc l'observation de la suite d'actions γ à travers le critère d'observation $\mathcal{O}_{\text{auth}}$ correspond à la seule action auth_{id_B}, d'où $\text{auth}_{id_B} \in \mathcal{O}_{\text{auth}}(\gamma)$. Ainsi, nous pouvons conclure que le processus $(Q \setminus \Gamma)/\mathcal{O}_{\text{auth}}$ peut effectuer l'action auth_{id_B}, mais pas le processus $(Q \setminus Act_E)/\mathcal{O}_{\text{auth}}$. Par conséquent, le processus $Q \setminus \Gamma$ n'est pas $\mathcal{O}_{\text{auth}}$-simulé par le processus $Q \setminus Act_E$, et nous pouvons conclure qu'ils ne sont donc pas $\mathcal{O}_{\text{auth}}$-bisimilaires.

Notons qu'une approche analogue basée sur la non interférence est proposée par Durante, Focardi & Gorrieri [38], et avec laquelle le protocole de Woo-Lam est analysé. Suite à cette analyse, les auteurs doivent filtrer manuellement les interférences qui ne correspondent pas à des attaques. Par exemple, leur méthode détecte la suite d'actions suivante

$$\begin{aligned}\gamma = \; & \overline{\delta^{c_1E}_{id_A}}(id_A)\ \delta^{c_1E}_{id_E}(id_A)\ \overline{\delta^{c_1B}_{id_E}}(id_A)\ \delta^{c_1B}_{id_B}(id_A)\ \mathrm{newNumber}_{id_B}\ \overline{\delta^{c_2A}_{id_B}}(n_B) \\ & \delta^{c_2A}_{id_E}(n_B)\ \overline{\delta^{c_2A}_{id_E}}(n_B)\ \delta^{c_2A}_{id_A}(n_B)\ \mathrm{enc}_{id_A}\ \overline{\delta^{c_3E}_{id_A}}(\{n_B\}_{k_{AS}}).\end{aligned}$$

Cependant, nous pouvons facilement remarquer que cette suite d'actions ne correspond pas à une attaque réussit du point de vue de l'authentification. L'interférence admissible permet donc une spécification explicite de ces attaques inoffensives. Ainsi, seulement les failles pouvant mener à de réelles attaques sont détectées à la suite d'une analyse du protocole. De même, en identifiant de telles interférences admissibles avant d'effectuer une analyse automatique du protocole de sécurité, nous pouvons bénéficier d'un gain au niveau de la précision et de la clarté des résultats obtenus de l'analyse.

7.4 Protocole de Needham-Schröder

Dans cette section, nous considérons le *protocole d'authentification à clé publique de Needham-Schröder* [86] qui se déroule comme suit :

$$\begin{array}{lcl} \text{Message 1 :} & A \xrightarrow{id_A, id_B, \{n_A, id_A\}_{k_B}} & B \\ \text{Message 2 :} & B \xrightarrow{id_B, id_A, \{n_A, n_B\}_{k_A}} & A \\ \text{Message 3 :} & A \xrightarrow{id_A, id_B, \{n_B\}_{k_B}} & B. \end{array}$$

(Consulter la Section 1.2.1 pour une description détaillée de ce protocole.)

Pour la suite de cette section, nous considérons la spécification du protocole de Needham-Schröder dans l'algèbre de processus SPPA donnée à l'Exemple 3.8. Rappelons que la spécification du participant A est telle que A peut initier le protocole avec n'importe quel autre participant. En particulier, étant donné un processus ennemi $E = (S_E, id_E)$, le participant A peut initier le protocole soit avec B, soit avec

E. Posons $P ::= A \parallel B$ le processus SPPA correspondant à cette spécification du protocole de Needham-Schröder. À partir de celle-ci, nous supposons que les actions critiques du participant associé à l'identificateur id sont exactement les appels de fonction auth_{id}, car ces actions sont seulement exécutées lorsque le participant en question (c'est-à-dire A ou B) est persuadé qu'il a authentifié l'autre. D'où, nous posons $Act_{\text{auth}} = \{\text{auth}_A, \text{auth}_B\}$.

7.4.1 Spécification de l'attaque sur le protocole de Needham-Schröder

Dans cette section, nous offrons une spécification de la fameuse attaque sur le protocole de Needham-Schröder, initialement découverte par Lowe [74]. Une description détaillée du déroulement de cette attaque est exposée à la Section 1.2.1. Notons que cette attaque de type « *man-in-the-middle* » est seulement possible lorsque le participant A initie le protocole avec un processus ennemi. Dans ce cas, ce dernier utilise l'information obtenue de A afin d'initier une exécution parallèle du protocole avec un autre participant dans le but de se faire passer pour A. Dans le contexte de SPPA, cette attaque correspond au processus ennemi $E = (S_E, id_E)$ dont l'agent initial S_E est donné par :

$$
\begin{aligned}
S_E ::= \; & c_{1E}(z_1).\ \texttt{let}\ (z_2, z_3, z_4) = z_1\ \texttt{in}\ \texttt{case}\ z_4\ \texttt{of}\ \{z_5\}_{k_E}\ \texttt{in} \\
& \texttt{let}\ z_6 = \text{enc}(k_B, z_5)\ \texttt{in}\ \texttt{let}\ z_7 = \text{pair}(z_2, id_B, z_6)\ \texttt{in} \\
& \overline{c_{1B}}(z_7).\ c_{2A}(z_8).\ \texttt{let}\ (z_9, z_{10}, z_{11}) = z_8\ \texttt{in} \\
& \texttt{let}\ z_{12} = \text{pair}(id_E, z_{10}, z_{11})\ \texttt{in} \\
& \overline{c_{2A}}(z_{12}).\ c_{3E}(z_{13}).\ \texttt{let}\ (z_{14}, z_{15}, z_{16}) = z_{13}\ \texttt{in} \\
& \texttt{case}\ z_{16}\ \texttt{of}\ \{z_{17}\}_{k_E}\ \texttt{in}\ \texttt{let}\ z_{18} = \text{enc}(k_B, z_{17})\ \texttt{in} \\
& \texttt{let}\ z_{19} = \text{pair}(z_2, id_B, z_{18})\ \texttt{in}\ \overline{c_{3B}}(z_{19}).\mathbf{0}
\end{aligned}
$$

À partir de ce processus ennemi, nous supposons que l'ensemble Γ des attaques admissibles est défini de façon analogue à celui du protocole de Woo-Lam (voir Section 7.3.3). Par conséquent, toute action de marquage d'entrée $\delta^c_{id_E}(a)$ et tout appel de fonction (cryptage enc_{id_E}, décryptage dec_{id_E}, couplage pair_{id_E} et authen-

tification auth_{id_E}) sont considérés comme étant des attaques admissibles car elles sont requises dès qu'un processus ennemi souhaite participer honnêtement au protocole. De plus, nous supposons que les actions de marquage de sortie suivantes correspondent à des attaques admissibles (pour tout $id \in \mathcal{I}$) :

1. $\overline{\delta^{c_{1id}}_{id_E}}(a)$ tel que $\models \exists_{x,y,z}\ (a == (id_E, x, \{(y, id_E)\}_z) \wedge \mathcal{I}(x) \wedge \mathcal{N}(y) \wedge \mathcal{K}(z))$;
2. $\overline{\delta^{c_{2id}}_{id_E}}(a)$ tel que $\models \exists_{x,y,z,w}\ (a == (id_E, x, \{(y, z)\}_w) \wedge \mathcal{I}(x) \wedge \mathcal{N}(y) \wedge \mathcal{N}(z) \wedge \mathcal{K}(w))$;
3. $\overline{\delta^{c_{3id}}_{id_E}}(a)$ tel que $\models \exists_{x,y,z}\ (a == (id_E, x, \{y\}_z) \wedge \mathcal{I}(x) \wedge \mathcal{N}(y) \wedge \mathcal{K}(z))$.

Ainsi, nous pouvons supposer que l'ensemble Γ des attaques admissibles contient toute action provenant de l'ensemble Act_E (les actions ennemies), à l'exceptions des marqueurs de sortie qui ne sont pas énumérés ci-dessus.

7.4.2 Analyse de flots d'information du protocole de Needham-Schröder

Par la Définition 6.8, et à l'aide du processus ennemi E défini plus haut, nous démontrons que le protocole d'authentification de Needham-Schröder ne préserve pas l'authenticité. En effet, il suffit de constater que

$$Q \setminus \Gamma \not\approx_{\mathcal{O}_{\text{auth}}} Q \setminus Act_E$$

pour un certain processus dérivé $Q \in \mathcal{D}(P_E)$. Soit $Q \in \mathcal{D}(P_E)$ le processus qui peut exécuter le calcul de la suite d'actions suivante :

$$\gamma = \overline{\delta^{c_{3B}}_{id_E}}(id_A, id_B, \{n_B\}_{k_B})\ \delta^{c_{3B}}_{id_B}(id_A, id_B, \{n_B\}_{k_B})\ \text{dec}_{id_B}\ \text{auth}_{id_B}.$$

Puisque $\overline{\delta^{c_{3B}}_{id_E}}(id_A, id_B, \{n_B\}_{k_B}) \in Act_E \setminus \Gamma$ et $\text{auth}_{id_B} \in Act_{\text{auth}}$, donc $\text{auth}_{id_B} \in \mathcal{O}_{\text{auth}}(\gamma)$, nous pouvons conclure que

$$(Q \setminus \Gamma)/\mathcal{O}_{\text{auth}} \xrightarrow{\text{auth}_{id_B}} Q'$$

pour un certain processus Q'. D'autre part, nous pouvons constater que le processus $(Q \setminus Act_E)/\mathcal{O}_{\text{auth}}$ ne peut pas exécuter l'action auth_{id_B} (car $\overline{\delta_{id_E}^{c_{3B}}}(id_A, id_B, \{n_B\}_{k_B}) \in Act_E \setminus \Gamma$). D'où, les processus $Q \setminus \Gamma$ et $Q \setminus Act_E$ ne sont pas $\mathcal{O}_{\text{auth}}$-bisimilaires. Notons que la dérivée Q provient de la branche du protocole P dans laquelle le participant A a initié une exécution du protocole avec le processus ennemi E. La suite d'actions γ correspond donc à une initiation frauduleuse du protocole par le processus E avec le participant B, dans le but de se faire passer pour A.

7.5 Transmission Control Protocol

Le protocole TCP (*Transmission Control Protocol*) permet d'établir un canal sécuritaire d'échange de données pour de diverses applications. Une connexion TCP utilise communément des structures de données afin d'entreposer certaines informations liées aux extrémités du lien de communication, l'état TCP actuel, les adresses IP, le numéro du port utilisé, la minuterie, le numéro d'identification de la session, la statut du flot de contrôle, etc. Une description complète du protocole TCP en terme de machine à états est offerte par Schuba et al. [101].

Avant de débuter la transmission de données entre une source A et une destination B, le protocole TCP utilise un protocole préliminaire, appelé *three-way handshake*, afin d'établir un lien antre A et B. Ce protocole se déroule comme suit :

$$\begin{array}{lcc} \text{Message 1 :} & A \xrightarrow{SYN_n} & B \\ \text{Message 2 :} & B \xrightarrow{SYN_m, ACK_n} & A \\ \text{Message 3 :} & A \xrightarrow{ACK_m} & B\ . \end{array}$$

Premièrement, le participant A initie la connexion en envoyant à B un paquet de synchronisation SYN_n, qui contient un numéro d'identification n, de même que les adresses IP de A et B (représentées par leurs identificateurs respectifs id_A et id_B). Ensuite, le participant B retourne à A un accusé de réception, représenté par le paquet ACK_n, et un nouveau paquet de synchronisation SYN_m. Finalement, A

retourne à B l'accusé de réception constitué du paquet ACK_m.

7.5.1 Spécification du protocole TCP

Une session du *three-way handshake* du protocole TCP est spécifié par le processus SPPA :

$$TCP_1 \ ::= \ A \parallel B_1$$

avec $C = \{c_1, c_2, c_3\}$ et où les participants $A = (S_A, id_A)$ et $B_1 = (S_{B_1}, id_B)$ sont définis comme suit :

$$\begin{aligned} S_A ::= \ & \texttt{let } x_1 = \text{newNumber}(-) \texttt{ in let } x_2 = \text{makeSYN}(id_A, id_B, x_1) \texttt{ in} \\ & \texttt{let } x_3 = \text{store}(x_2) \texttt{ in } \overline{c_1}(x_2).\ c_2(x_4).\ \texttt{let } (x_5, x_6) = x_4 \texttt{ in} \\ & \texttt{let } x_7 = \text{makeACK}(x_2) \texttt{ in } [x_7 = x_6] \texttt{ let } x_8 = \text{makeACK}(x_5) \texttt{ in} \\ & \overline{c_3}(x_8).S'_A \end{aligned}$$

$$\begin{aligned} S_{B_1} ::= \ & c_1(y_1).\ \texttt{let } (y_2, y_3, y_4, y_5) = y_1 \texttt{ in } [y_4 = id_B] \texttt{ let } y_6 = \text{store}(y_1) \texttt{ in} \\ & \texttt{let } y_7 = \text{makeACK}(y_1) \texttt{ in let } y_8 = \text{newNumber}(-) \texttt{ in} \\ & \texttt{let } y_9 = \text{makeSYN}(id_B, y_3, y_8) \texttt{ in let } y_{10} = \text{store}(y_9) \texttt{ in} \\ & \texttt{let } y_{11} = \text{pair}(y_9, y_7) \texttt{ in} \\ & \overline{c_2}(y_{11}).\ c_3(y_{12}).\ \texttt{let } y_{13} = \text{makeACK}(y_9) \texttt{ in } [y_{13} = y_{12}]\ S'_B \end{aligned}$$

où S'_A and S'_B sont des agents qui exécutent le reste du protocole TCP.

Notons que les paquets de synchronisation SYN_n et d'accusé de réception ACK_n correspondent formellement à des tuplets $(header, id_X, id_{X'}, n)$ de notre algèbre de message où $id_X, id_{X'} \in \mathcal{I}$ sont les identificateurs respectifs de la source et de la destination, et $n \in \mathcal{N}$ est le numéro d'identification de la session TCP. Pour les besoins de notre spécification, ces paquets sont formalisés par des 4-tuples $SYN_n = (0, id, id', n)$ et $ACK_n = (1, id, id', n)$ (l'en-tête « 0 » désigne un paquet SYN, tandis que l'en-tête « 1 » désigne un paquet ACK, bien qu'en pratique ces en-têtes sont plus longues car elles contiennent d'autres informations relatives l'instance du protocole). Cette spécification nécessite les fonctions suivantes :

- makeSYN$(id, id', n) = (0, id, id', n)$ est un *constructeur de paquets de synchronisation* $\phi_{\text{makeSYN}} ::= \mathcal{I}(x_1) \wedge \mathcal{I}(x_2) \wedge \mathcal{N}(x_3)$;
- store$(a) = a$ est une *fonction d'entreposage* avec $\phi_{\text{store}} ::= \mathcal{M}(x)$;
- makeACK$(0, id, id', n) = (1, id, id', n)$ est une *fonction d'accusé de réception* avec $\phi_{\text{makeACK}} ::= (x_1 == 0) \wedge \mathcal{I}(x_2) \wedge \mathcal{I}(x_3) \wedge \mathcal{N}(x_4)$;

La valeur retournée par la fonction store(a) désigne, intuitivement, la référence (adresse locale) où le message a est entreposé. Puisque le protocole TCP requiert aucune opération cryptographique, notre spécification utilise l'ensemble de fonctions suivant :

$$\mathcal{F} = \{\text{pair}, \text{newNumber}, \text{makeSYN}, \text{store}, \text{makeACK}\}.$$

La participant B, communément appelé le serveur, peut généralement traiter plusieurs demandes de branchement simultanées. Supposons que N désigne le nombre maximal de connexions TCP actives permises par B (avant une réinitialisation). Dans ce cas, le serveur B est spécifié par le participant $B = (S_B, id_B)$ avec $S_B ::= S_{B_1} | \ldots | S_{B_1}$ (N fois). Ainsi, le protocole TCP entier, qui peut traiter plusieurs sessions simultanément, est spécifié par le processus SPPA

$$TCP ::= A \parallel B.$$

Nous utiliserons cette spécification pour le reste de cette section.

7.5.2 Spécification de l'attaque de SYN flooding

L'attaque de *SYN flooding* [101] sur le protocole TCP est certainement l'une des plus célèbres attaques de DoS. Celle-ci fut perpétrée sur plusieurs sites Internet à caractère commercial depuis son introduction en 1996. L'attaque de *SYN flooding* permet à un intrus de provoquer un DoS par épuisement de ressource chez un serveur en initiant, avec très peu d'efforts, un grand nombre de sessions du protocole TCP avec la victime. Cette faille dans le protocole TCP est principalement due à

la facilité de forger une identité (adresse IP) et, par conséquent, la difficulté pour la victime d'identifier proprement un intrus.

Plus précisément, l'attaque d'épuisement de ressource *SYN flooding* sur le protocole TCP se déroule comme suit :

1. newId_{id_E} : l'intrus forge un identificateur id (adresse IP) ;
2. newNumber_{id_E} : l'intrus génère un nombre aléatoire n ;
3. makeSYN_{id_E} : l'intrus construit un paquet de synchronisation SYN_n qui contient n comme numéro d'identification, l'adresse IP de la victime (représentée par l'identificateur id_B) et une adresse IP source forgée (représentée par l'identificateur forgé id) ;
4. $\overline{\delta^{c_1}_{id_E}}(SYN_n)$: l'intrus envoie le paquet de synchronisation SYN_n à sa victime (le serveur B).

Suite à la réception du paquet de synchronisation de l'intrus, le serveur se comporte comme suit :

1. store_{id_B} : le serveur alloue des structures de données afin d'entreposer le paquet de synchronisation SYN_n ;
2. makeACK_{id_B} : le serveur crée un paquet d'accusé de réception ACK_n correspondant à SYN_n ;
3. newNumber_{id_B} : le serveur génère un nombre aléatoire m ;
4. makeSYN_{id_B} : le serveur construit un nouveau paquet de synchronisation SYN_m ;
5. $\overline{\delta^{c_2}_{id_B}}(SYN_m, ACK_n)$: le serveur envoie les paquets SYN_m et ACK_n à E.

Évidemment, l'intrus ignore ce dernier message envoyé par le serveur. (En fait, ce message ne devrait se rendre nul part puisque le serveur l'envoie à l'adresse IP forgée par l'intrus.) L'intrus va plutôt répéter cette attaque avec différents identificateurs forgés et différents paquets de synchronisation SYN_n, mais sans jamais compléter le protocole.

L'attaque de *SYN flooding* est donc possible puisque la génération d'identificateurs forgés et de paquets de synchronisation nécessite peu de ressources. Or, un DoS par épuisement de ressource se produit car le serveur alloue des structures de données coûteuses lors de la réception d'un paquet de synchronisation (formalisé ci-dessus par l'action store_{id_B}). Afin d'analyser cette attaque de DoS, nous devons donc comparer le coût pour l'intrus de perpétrer son attaque et le coût pour le serveur de traiter celle-ci. Si le coût de l'attaque (c'est-à-dire $\text{coût(newId)} + \text{coût(newNumber)} + \text{coût(makeSYN)} + \text{coût}(\overline{\delta^{c_1}_{id_E}})$) est largement inférieur au coût pour la traiter ($\text{coût(store)} + \text{coût(makeACK)} + \text{coût(newId)} + \text{coût(makeSYN)} + \text{coût}(\overline{\delta^{c_2}_{id_B}})$), alors nous pouvons conclure que le protocole admet une faille. En effet, un intrus pourrait initier, à l'aide de très peu de ressources, une multitude d'attaques qui provoqueront chez le serveur un important gaspillage de ressources. Si cette dépense de ressources est au-delà des capacités du serveur, alors le serveur n'aura plus de ressources disponibles et devra refuser toute nouvelle demande de branchement, incluant les demandes honnêtes.

À partir de la spécification du protocole TCP donnée ci-dessus, nous considérons le processus ennemi $E = (S_E, id_E)$ qui exécute l'attaque de *SYN flooding*, où l'agent initial S_E est donné par :

$$S_E ::= E_1 | \ldots | E_1 \ (N \text{ fois})$$

où S_{E_1} est l'agent qui exécute une seule attaque, défini par :

$$\begin{aligned} S_E ::= \ & \texttt{let } z_1 = \text{newId}(-) \texttt{ in let } z_2 = \text{newNumber}(-) \texttt{ in} \\ & \texttt{let } z_3 = \text{makeSYN}(z_1, id_B, z_2) \texttt{ in } \overline{c_1}(z_3).\mathbf{0} \end{aligned}$$

Dans cette spécification, l'intrus envoie à B un paquet de synchronisation SYN_n contenant un identificateur forgé id à la place de celui de E.

L'ensemble Γ des attaques admissibles est composé des actions de marquage de sortie et d'entrée dans lesquelles les paquets de synchronisation et d'accusé de réception contiennent id_E comme adresse IP source ou de destination (correspondant à une instance honnête du protocole). Formellement, nous avons

$$\begin{aligned}\Gamma = \; & \{\overline{\delta^{c_1}_{id_E}}(0, id_E, id, n),\ \delta^{c_1}_{id_E}(0, id, id_E, n) \mid id \in \mathcal{I} \text{ et } n \in \mathcal{N}\} \\ & \cup \{\overline{\delta^{c_2}_{id_E}}(SYN^1_m, ACK^1_n),\ \delta^{c_2}_{id_E}(SYN^2_m, ACK^2_n) \mid id \in \mathcal{I} \text{ et } m, n \in \mathcal{N}\} \\ & \cup \{\overline{\delta^{c_3}_{id_E}}(1, id, id_E, m),\ \delta^{c_3}_{id_E}(1, id_E, id, m) \mid id \in \mathcal{I} \text{ et } m \in \mathcal{N}\}\end{aligned}$$

avec $SYN^1_m = (0, id_E, id, m)$, $ACK^1_n = (1, id_E, id, n)$, $SYN^2_m = (0, id, id_E, m)$ et $ACK^2_n = (1, id_E, id, n)$.

L'attaque de *SYN flooding* est donc formalisé par le processus SPPA

$$TCP_E \; ::= \; E \parallel B$$

où B est le participant SPPA défini plus haut (voir Section 7.5.1).

À partir de notre spécification du protocole TCP, nous pouvons établir la distribution suivante des coûts reliés à son exécution. Tout d'abord, nous supposons que les actions makeACK_{id} (de même que les actions d'échec correspondantes $\texttt{fail}^{\text{makeACK}}_{\texttt{id}}$) sont celles qui ont un coût CPU le plus élevé, bien que nous supposons que leur coût CPU ne dépasse pas la capacité CPU de B, ni celle de E. D'où

$$\rho_{\text{cpu}}(\alpha) < \text{CPU}_B \quad \text{et} \quad \rho_{\text{cpu}}(\alpha) < \text{CPU}_E$$

pour toute action $\alpha \in Act$. Cependant, nous supposons que les actions d'entreposage store_{id} sont les seules actions qui ont un coût mémoire qui excèdent la capacité mémoire de B, ainsi que celle de E. D'où,

$$\rho_{\text{mem}}(\text{store}_{id_B}) > \text{MEM}_B \quad \text{et} \quad \rho_{\text{mem}}(\text{store}_{id_E}) > \text{MEM}_E.$$

Par conséquent, le serveur peut subir un épuisement de ressource mémoire seulement lorsqu'il entrepose des données (ce qui est dû à la création de structures de données dispendieuses). D'autre part, nous pouvons facilement remarquer que le processus ennemi E défini plus haut respecte ses capacités car il n'exécute aucune action d'entreposage.

À l'aide de ce processus ennemi et de la Définition 6.9, nous pouvons conclure

que le protocole TCP n'est pas impassible. De façon plus précise, nous voyons que

$$TCP_E \setminus \Gamma \not\approx_{\mathcal{O}_{\text{coûteux}}} TCP_E \setminus Act_E.$$

En effet, le processus $TCP_E ::= E \parallel B$ peut effectuer calcul de la suite d'actions suivante :

$$\begin{aligned}\gamma \;=\; & \text{newId}_{id_E}\ \text{newNumber}_{id_E}\ \text{makeSYN}_{id_E}\ \overline{\delta^{c_1}_{id_E}}(SYN_n)\ \delta^{c_1}_{id_B}(SYN_n) \\ & \text{split}_{id_B}\ \text{store}_{id_B}\end{aligned}$$

avec $SYN_n = (0, id, id_B, n)$ pour certain identificateur $id \in \mathcal{I}$ un certain nombre $n \in \mathcal{N}$. Donc $\overline{\delta^{c_1}_{id_E}}(SYN_n) \notin \Gamma$. Or, puisque $\text{store}_{id_B} \in \mathcal{O}_{\text{coûteux}}(\gamma)$ et γ ne contient aucune action provenant de l'ensemble Γ, nous pouvons constater que le processus $(TCP_E \setminus \Gamma)/\mathcal{O}_{\text{coûteux}}$ peut exécuter l'action store_{id_B}, mais ce n'est pas le cas pour le processus $(TCP_E \setminus Act_E)/\mathcal{O}_{\text{coûteux}}$ car γ est contient des actions de l'ensemble Act_E. Nous pouvons donc conclure que ces deux processus ne sont pas bisimilaires.

7.6 Protocole de paiement électronique sécuritaire 1KP

La famille de protocoles de paiement électronique sécuritaire iKP (pour $i = 1, 2, 3$) furent développés par la Division Recherche de IBM [16,17]. Ces protocoles offrent un service de transaction par carte de crédit entre un acquérant et un marchand par l'entremise des réseaux financiers existants pour la procédure de certification et d'autorisation. Les trois protocoles iKP sont composés des mêmes étapes et ils diffèrent l'un de l'autre seulement au niveau du contenu des messages envoyés à chaque étape. Ces protocoles utilisent le cryptage à clé publique et la signature à clé publique, de même qu'une fonction de hachage fortement résistante aux collisions.

Dans ce qui suit, nous désignons respectivement par pk_A et sk_A la clé publique et la clé privée d'un participant A (avec $pk_A^{-1} = sk_A$). Pour les besoins de ce travail, nous étudions uniquement le protocole 1KP, le plus simple du trio, qui se déroule comme suit :

$$
\begin{array}{lcc}
\text{Message 1 :} & A & \xrightarrow{salt_A, h(r_A, ban_A)} \quad B \\
\text{Message 2 :} & B & \xrightarrow{clear} \quad A \\
\text{Message 3 :} & A & \xrightarrow{\{slip\}_{pk_{ACQ}}} \quad B \\
\text{Message 4 :} & B & \xrightarrow{clear, h(salt_A, desc), \{slip\}_{pk_{ACQ}}} \quad ACQ \\
\text{Message 5 :} & ACQ & \xrightarrow{resp, [resp, h(common)]_{sk_{ACQ}}} \quad B \\
\text{Message 6 :} & B & \xrightarrow{resp, [resp, h(common)]_{sk_{ACQ}}} \quad A
\end{array}
$$

où

$$common = (price, id_B, tid_B, n_B, h(r_A, ban_A), h(salt_A, desc)),$$

$$clear = (id_B, tid_B, n_B, h(common))$$

et

$$slip = (price, h(common), ban_A, r_A, exp_A).$$

Tout d'abord, l'acquérant A envoie au marchand B un nombre aléatoire $salt_A$ et un second nombre aléatoire r_A haché avec son numéro de compte ban_A (son numéro de carte de crédit, par exemple). Ensuite, le marchand B répond à A en lui envoyant son identificateur id_B, le numéro d'identification de la transaction tid_B, un nonce n_B fraîchement généré, et la valeur hachée du champ *common*, où *price* et *desc* correspondent respectivement au prix et à la description de l'achat. Troisièmement, l'acquérant A envoie à B le champ *slip* chiffré avec la clé publique pk_{ACQ} de l'autorité de certification ACQ (la banque, par exemple), où exp_A désigne la date d'expiration associée au numéro de compte ban_A. Le marchand B peut ensuite entamer la procédure de certification auprès de ACQ afin d'obtenir l'autorisation de celui-ci avant de conclure la transaction. Pour ce, B envoie à l'autorité de certification ACQ le message fraîchement reçu de A accompagné des champs *clear* et $h(salt_A, desc)$. Suite à la réception de ce message, l'autorité de certification ACQ procède comme suit :

1. ACQ s'assure d'abord que les valeurs incluses dans le champ *clear* ne furent pas préalablement utilisées ;

2. ACQ décrypte $\{slip\}_{pk_{ACQ}}$ et obtient $slip$;
3. ACQ vérifie si le $h(common)$ du champ $clear$ correspond bien à celui du champ $slip$;
4. ACQ reconstruit le champ $common$, calcul $h(common)$ et vérifie si cette valeur correspond bien à celle contenue dans le champ $clear$;
5. ACQ vérifie si les valeurs ban_A et exp_A sont valides, par l'entremise du système de certification et d'autorisation fourni par l'institution financière émettrice du compte (ou de la carte de crédit), et obtient une autorisation (ou un refus) en-ligne pour le paiement.

Une fois cette procédure d'autorisation terminée, l'autorité de certification ACQ envoie au marchand B la réponse $resp$ obtenue du système de certification (« accepté » ou « refusé ») accompagnée de $h(common)$ et $resp$ signés ensemble avec la clé privée de ACQ. Finalement, le marchand B vérifie la signature de ACQ et transfert le message reçu de ACQ à l'acquérant A.

7.6.1 Spécification du protocole 1KP

Afin de spécifier le protocole 1KP dans l'algèbre de processus SPPA, nous considérons les participants SPPA $A = (S_A, id_A)$, $B = (S_B, id_B)$ et $ACQ = (S_{ACQ}, id_{ACQ})$, dont les agents initiaux sont définis à la Figure 7.1.

Notre spécification du protocole 1KP utilise l'ensemble de fonctions

$$\mathcal{F} = \{\text{pair, hash, enc, sign, replay, clearing}\}.$$

Notons que cette spécification nécessite deux fonctions de hachage, l'une avec un seul argument et une seconde avec deux arguments, mais, par souci de simplicité, nous utilisons le même nom pour désigner ces deux fonctions. D'autre part, la fonction replay$(id, tid, n, h(a))$ permet à l'autorité de certification de s'assurer que les valeurs du champ $clear$ ne furent pas préalablement utilisées; ainsi elle retourne 1 ($vrai$) si c'est le cas, et produit un échec sinon. D'où, le domaine de replay est donc

$$
\begin{array}{l}
S_A ::= \ \texttt{let}\ x_1 = \text{newNumber}(-)\ \texttt{in}\ \texttt{let}\ x_2 = \text{pair}(x_1, ban_A)\ \texttt{in} \\
\quad \texttt{let}\ x_3 = h(x_2)\ \texttt{in}\ \texttt{let}\ x_4 = \text{newNumber}(-)\ \texttt{in} \\
\quad\quad \texttt{let}\ x_5 = \text{pair}(x_4, x_3)\ \texttt{in} \\
\quad\quad\quad \overline{c_1}(x_5).\ c_2(x_6).\ \texttt{let}\ (x_7, x_8, x_9, x_{10}) = x_6\ \texttt{in} \\
\quad\quad\quad\quad \texttt{let}\ x_{11} = \text{pair}(x_4, desc)\ \texttt{in}\ \texttt{let}\ x_{12} = h(x_4, desc)\ \texttt{in} \\
\quad\quad\quad\quad\quad \texttt{let}\ x_{13} = \text{pair}(price, x_7, x_8, x_9, x_3, x_{12})\ \texttt{in} \\
\quad\quad\quad\quad\quad \texttt{let}\ x_{14} = h(x_{13})\ \texttt{in}\ [x_{14} = x_{10}] \\
\quad\quad\quad\quad\quad\quad \texttt{let}\ x_{15} = \text{pair}(price, x_{14}, ban_A, x_1, exp_A)\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad\quad \texttt{let}\ x_{16} = \text{enc}(pk_{ACQ}, x_{15})\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad\quad\quad \overline{c_3}(x_{16}).\ c_6(x_{17}).\ \texttt{let}\ (x_{18}, x_{19}) = x_{17}\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \texttt{let}\ x_{20} = \text{pair}(x_{18}, x_1 0)\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \texttt{case}\ x_{19}\ \texttt{of}\ [x_{20}]_{sk_{ACQ}}\ \texttt{in}\ \mathbf{0}
\end{array}
$$

$$
\begin{array}{l}
S_B ::= \ c_1(y_1).\texttt{let}\ (y_2, y_3) = y_1\ \texttt{in}\ \texttt{let}\ y_4 = h(y_2, desc)\ \texttt{in} \\
\quad \texttt{let}\ y_5 = \text{newNumber}(-)\ \texttt{in} \\
\quad\quad \texttt{let}\ y_6 = \text{pair}(price, id_B, tid_B, y_5, y_3, y_4)\ \texttt{in} \\
\quad\quad\quad \texttt{let}\ y_7 = h(y_6)\ \texttt{in}\ \texttt{let}\ y_8 = (id_B, tid_B, y_5, y_7)\ \texttt{in} \\
\quad\quad\quad\quad \overline{c_2}(y_8).\ c_3(y_9).\ \texttt{let}\ y_{10} = \text{pair}(y_8, y_4, y_9)\ \texttt{in} \\
\quad\quad\quad\quad\quad \overline{c_4}(y_{10}).\ c_5(y_{11}).\ \texttt{let}\ (y_{12}, y_{13}) = y_{11}\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad \texttt{let}\ y_{14} = \text{pair}(y_1 2, y_7)\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad\quad \texttt{case}\ y_{13}\ \texttt{of}\ [y_{14}]_{sk_{ACQ}}\ \texttt{in}\ \overline{c_6}(y_{11}).\mathbf{0}
\end{array}
$$

$$
\begin{array}{l}
S_{ACQ} ::= \ c_4(z_1).\texttt{let}\ (z_{11}, z_{12}, z_{13}) = z_1\ \texttt{in} \\
\quad \texttt{let}\ (z_{111}, z_{112}, z_{113}, z_{114}, z_{115}) = z_{11}\ \texttt{in} \\
\quad\quad \texttt{let}\ z_2 = \text{replay}(z_{111}, z_{112}, z_{113}, z_{114})\ \texttt{in}\ [z_2 = 1] \\
\quad\quad\quad \texttt{case}\ z_{13}\ \texttt{of}\ \{z_3\}_{sk_{ACQ}}\ \texttt{in} \\
\quad\quad\quad \texttt{let}\ (z_{31}, z_{32}, z_{33}, z_{34}, z_{35}) = z_3\ \texttt{in} \\
\quad\quad\quad\quad [z_{115} = z_{32}]\ \texttt{let}\ z_4 = h(z_{34}, z_{33})\ \texttt{in} \\
\quad\quad\quad\quad\quad \texttt{let}\ z_5 = (z_{31}, z_{111}, z_{112}, z_{113}, z_{114}, z_4, z_{12})\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad \texttt{let}\ z_6 = h(z_5)\ \texttt{in}[z_7 = z_{32}] \\
\quad\quad\quad\quad\quad\quad \texttt{let}\ z_7 = (z_{33}, z_{35}, z_{31})\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad \texttt{let}\ z_8 = \text{clearing}_{id_{ACQ}}(z_7)\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad\quad \texttt{let}\ z_9 = (z_8, z_6)\ \texttt{in}\ \texttt{let}\ w_1 = [z_9]_{sk_{ACQ}}\ \texttt{in} \\
\quad\quad\quad\quad\quad\quad\quad \texttt{let}\ w_2 = (z_9, w_1)\ \texttt{in}\ \overline{c_5}(w_2).ACQ
\end{array}
$$

FIG. 7.1 – Spécification des participants du protocole 1KP.

uniquement composé des valeurs qui n'ont pas été utilisées dans une transaction précédente, et ce domaine dépend donc des messages entreposés par l'autorité de certification à un instant donné. De plus, nous supposons que la fonction replay entrepose son message d'entrée, et nécessite donc une certaine quantité de ressource mémoire. La fonction clearing$(ban, exp, price) = resp$ permet de valider les messages $ban, exp, price$ de façon à obtenir une autorisation pour le paiement. Elle retourne le message $resp = ok$ (ou 1) si la validation est réussite, et le message $resp = nok$ (ou 0) autrement.

Nous supposons que les actions de signature sign_{id}, de vérification de signature signv_{id} et de validation clearing_{id} sont les plus coûteuses du point de vue des ressources CPU. Ainsi, nous supposons que ces actions excèdent la capacité CPU des participants honnêtes, c'est-à-dire $\rho_{\text{cpu}}(\alpha) > \text{CPU}_B$, CPU_{ACQ} dès que

$$\alpha \in \{ \text{signv}_{id_B}, \text{sign}_{id_B}, \text{signv}_{id_{ACQ}}, \text{sign}_{id_{ACQ}}, \text{clearing}_{id_{ACQ}} \\ \texttt{fail}^{\text{signv}}_{\texttt{id}_\texttt{B}}, \texttt{fail}^{\text{sign}}_{\texttt{id}_\texttt{B}}, \texttt{fail}^{\text{signv}}_{\texttt{id}_\texttt{ACQ}}, \texttt{fail}^{\text{sign}}_{\texttt{id}_\texttt{ACQ}}, \texttt{fail}^{\text{clearing}}_{\texttt{id}_\texttt{ACQ}} \}.$$

D'autre part, nous supposons que l'intrus possède une capacité CPU moindre que celles des autres participants. Plus spécifiquement, nous interdisons à un processus ennemi qui respecte ses capacités d'exécuter toute action de hachage, de cryptage, de décryptage, de signature et de vérification de signature. D'où, nous supposons que $\rho_{\text{cpu}}(\alpha) > \text{CPU}_E$ dès que

$$\alpha \in \{ \text{hash}_{id_E}, \text{enc}_{id_E}, \text{dec}_{id_E}, \text{sign}_{id_E}, \text{signv}_{id_E}, \\ \texttt{fail}^{\text{hash}}_{\texttt{id}_\texttt{E}}, \texttt{fail}^{\text{enc}}_{\texttt{id}_\texttt{E}}, \texttt{fail}^{\text{dec}}_{\texttt{id}_\texttt{E}}, \texttt{fail}^{\text{sign}}_{\texttt{id}_\texttt{E}}, \texttt{fail}^{\text{signv}}_{\texttt{id}_\texttt{E}} \}.$$

(Notons que cet ensemble contient aussi les actions d'échec correspondantes aux actions.) Pour ce qui est des coûts en ressources mémoire, seule l'action $\text{replay}_{id_{ACQ}}$ nécessite une certaine quantité de ressources pour entreposer des données. Ainsi, nous supposons que c'est l'unique action qui excède la capacité mémoire des participants, c'est-à-dire

$$\rho_{\text{mem}}(\text{replay}_{id_{ACQ}}) > \text{MEM}_{ACQ} \quad \text{et} \quad \rho_{\text{mem}}(\text{replay}_{id_E}) > \text{MEM}_E.$$

7.6.2 Attaque de déni de service sur le protocole 1KP

Considérons le processus ennemi $E = (S_E, id_E)$, avec

$$S_E ::= \texttt{let } x_1 = \text{newMessage}(-) \texttt{ in let } x_2 = \text{pair}(x_1, x_1) \texttt{ in}$$
$$\overline{c_1}(x_2).\ c_2(x_3).\ \overline{c_3}(x_1).\ c_4(x_4).\ \overline{c_5}(x_2).\mathbf{0}\ .$$

Nous pouvons facilement remarquer que ce processus ennemi respecte les capacités CPU et mémoire définies plus haut. Nous utilisons ce processus ennemi afin de démontrer que le protocole 1KP ne satisfait pas l'impassibilité, c'est-à-dire

$$P_E \setminus \Gamma \not\approx_{\mathcal{O}_{\text{coûteux}}} P_E \setminus Act_E.$$

En effet, E peut perpétrer l'attaque de DoS suivante simplement à partir d'un message forgé a. Cette attaque se déroule comme suit :

Message 1 :	E	$\xrightarrow{a,a}$	B
Message 2 :	B	$\xrightarrow{clear}$	E
Message 3 :	E	$\xrightarrow{a}$	B
Message 4 :	B	$\xrightarrow{clear,h(salt_A,desc),\{slip\}_{pk_{ACQ}}}$	$E\ (ACQ)$
Message 5 :	$E\ (ACQ)$	$\xrightarrow{a,a}$	B

D'abord, E envoie le couple de messages forgés (a, a) au marchand B. Celui-ci traite les données reçues (sans s'apercevoir qu'elles sont incorrectes) et répond à E. Suite à la réception de cette réponse, E envoie le message forgé a à B, qui le traitera encore sans soupçonner une attaque. Ainsi, le marchand B amorcera la procédure de validation en envoyant un message à l'autorité de certification ACQ par l'entremise du canal public c_4. Le processus ennemi E intercepte ce message (avant que ACQ puisse le recevoir) et retourne à B, en se faisant passer pour ACQ, un autre message forgé (a, a). Après avoir reçu ce message, B exécute une vérification de signature coûteuse afin de s'assurer de l'authenticité ce dernier message (cette vérification va évidemment échouer). Ainsi, le processus E réussi à provoquer l'exécution d'une action coûteuse par le marchand B. Si E exploite cette faille à plusieurs reprises, c'est-à-dire en initiant plusieurs fois le protocole 1KP avec un même marchand

B, que nous supposons capable de traiter simultanément un grand nombre de transactions, alors E pourrait causer chez B un DoS par épuisement de ressources CPU (dû à plusieurs vérifications de signature).

Cette attaque de DoS est détectée par notre propriété d'impassibilité car l'action ennemie $\overline{\delta^{c_1}_{id_E}}(a,a)$ (qui ne correspond pas à une attaque admissible) cause de l'interférence sur l'action d'échec de vérification de signature $\mathtt{fail}^{\mathrm{signv}}_{\mathtt{id_B}}$ (qui provient de la vérification « `case` a `of` $[z_2]_{sk_{ACQ}}$ `in` » effectuée par B). Puisque cette action excède la capacité CPU du marchand B, d'où $\mathtt{fail}^{\mathrm{signv}}_{\mathtt{id_B}} \in \mathtt{Act}_{\text{coûteux}}$, et que cette action ne se produira pas dans le processus $P_E \setminus Act_E$ (car $\overline{\delta^{c_1}_{id_E}}(a,a) \in Act_E$), nous pouvons donc conclure que $P_E \setminus \Gamma \not\approx_{\mathcal{O}_{\text{coûteux}}} P_E \setminus Act_E$. D'où, le protocole 1KP n'est pas impassible.

CHAPITRE 8

CONCLUSION

8.1 Principales contributions de la thèse

L'une des principales contributions de cette thèse est l'introduction de l'algèbre de processus SPPA qui étend CCS par des appels de fonctions explicites sans avoir à intégrer des règles de déductions, comme c'est le cas dans le spi calcul, qui sont souvent coûteuses d'un point de vue algorithmique. D'autre part, un opérateur d'observation est directement intégré à la syntaxe de SPPA. La sémantique de cet opérateur nous permet de décrire, entre autres, l'observation d'un protocole par chacun des ses participants : un tel participant ne peut observer que ses propres appels de fonction, de même que les messages qu'il envoie et reçoit (modélisés par des actions de marquage). De cette sémantique pour l'observation d'un processus nous obtenons la notion de bisimulation par observation qui nous procure une méthode naturelle pour comparer deux processus par rapport à un même critère d'observation. Cette relation d'équivalence est fondamentale à la définition et la vérification de la propriété d'interférence admissible. Nous introduisons aussi une extension symbolique de SPPA (voir Chapitre 4) qui s'obtient en assignant à chaque processus une formule spécifiant un ensemble de contraintes sur les variables libres du processus. Les formules utilisées proviennent d'une logique, que nous démontrons décidable, dont les termes atomiques sont ceux de l'algèbre de messages de SPPA. Nous présentons une sémantique symbolique pour ces processus contraints (voir Section 4.4). De plus, nous introduisons une relation de bisimulation pour les processus contraints (voir Section 4.6), et nous démontrons que celle-ci coïncide avec la relation de bisimulation lorsque restreinte aux processus (non symboliques) SPPA.

La contribution la plus importante de cette thèse est la formalisation de la propriété d'interférence admissible BNAI dans le contexte de SPPA (voir Section 5.2).

Nous présentons une méthode de preuve cohérente et complète pour la vérification de BNAI basée sur l'équivalence de bisimulation. Nous démontrons aussi que BNAI satisfait certaines propriétés de compositionalité par rapport aux principaux opérateurs de SPPA (voir Section 5.3). D'autre part, nous démontrons que plusieurs propriétés de non interférence, dont BNAI, ne sont pas définissables dans le μ-calcul (voir Section 5.4), ce qui explique que l'on doive les vérifier par une famille de bisimulations entre contextes.

Afin de motiver l'aspect pratique de BNAI, nous présentons dans cette thèse diverses applications à la validation de protocoles de sécurité. Tout d'abord, nous illustrons comment utiliser BNAI afin de vérifier si un protocole préserve la confidentialité des données secrètes de ces participants (voir Section 6.1). Puis, nous illustrons comment utiliser BNAI afin de vérifier si un protocole préserve l'authenticité de ses participants (voir Section 6.2). Finalement, nous introduisons une nouvelle méthode de validation qui permet de déterminer si un protocole de sécurité est vulnérable aux attaques de déni de service (voir Section 6.3). Cette application est développée dans une extension de SPPA à l'aide d'un modèle de coûts qui décrit les quantités de ressources mémoire et CPU utilisées par un participant au protocole. À partir de ce modèle, nous proposons une interprétation de BNAI qui nous permet de détecter les comportements qui sont susceptibles de provoquer un épuisement de ressources de l'un des participants du protocole lorsque celui-ci est attaqué par un intrus. Ces trois interprétations de BNAI sont appuyées par des études de cas complètes portant sur des protocoles connus (voir Chapitre 7).

8.2 Comparaisons avec les travaux voisins

La différence majeure entre notre approche basée sur une formalisation des propriétés de sécurité en termes d'interférence admissible et une approche basée sur une formalisation en terme de problème d'atteignabilité se résume de la façon suivante. Les deux méthodes détectent des traces qui correspondent à des attaques sur le protocole. Cependant, l'utilisation de non interférence nous permet d'établir

une dépendance causale entre les différents types d'actions (par exemple, le comportement de l'intrus et les actions critiques d'un participant). Ainsi, la méthode de validation présentée dans cette thèse, comparativement aux méthodes de traces, nous permet non seulement de trouver les attaques, mais aussi de prouver que celles-ci correspondent bien à des attaques.

Comme nous l'avons souligné dans cette thèse, la validation d'un protocole par rapport à tout environnement hostile est généralement indécidable due à la quantité de stratégies d'attaque possibles. Les méthodes de validation par *model-checking* basées sur CSP proposées par Lowe [74] et par Schneider [100] nécessitent la modélisation d'un intrus spécifique dont la définition dépend du protocole sous étude. La méthode proposée dans cette thèse est plutôt basée sur des propriétés de sécurité qui offrent une protection contre tous les processus ennemies définissables dans SPPA. Cependant, d'un point de vue pratique, et tout particulièrement pour une analyse de la robustesse contre le déni de service, nous vérifions seulement par rapport à un seul processus ennemi qui est suffisamment fort. Nous n'obtenons ainsi qu'une approximation de notre propriété de sécurité, ce qui nous ramène aux approches de Lowe et Schneider. Les approches basées sur le spi calcul développées par Abadi & Gordon [5] et Boreale, De Nicola & Pugliese [21] bénéficient de méthodes de validation par *equivalence-checking* qui prennent avantage d'une caractérisation finie de l'univers de tous les processus ennemis. Notons cependant que ces approches ne permettent qu'une formalisation de la sécurité qu'en termes de traces, et non en termes de flots d'information. D'autre part, Boreale et al. [21] formalisent la confidentialité d'un message m par une équivalence barbelée de tous les processus obtenus en substituant toute occurrence de m par d'autres messages. Ainsi, leur formalisation de la confidentialité garantit une indistinguabilité du protocole par rapport aux message secrets échangés. Cependant, leur formalisation, contrairement à la notre, ne considère pas comme un bris de confidentialité la divulgation de l'existence d'un message secret. En effet, notre formalisation de la confidentialité (Définition 6.6) garantit une indistinguabilité entre le protocole et le protocole dans lequel aucun message secret n'est échangé. Nous pouvons donc

conclure que notre formalisation raffine celle de Boréale et al.

Du point de vue de la modélisation, comparativement à d'autres algèbres de processus comme CCS, CSP et SPA, l'algèbre de processus SPPA permet une modélisation plus détaillée du comportement de l'intrus et des participants (grâce aux appels de fonctions) et une analyse plus poussée des échanges de messages entre les participants et l'intrus (grâce aux actions de marquage). En particulier, l'utilisation de SPPA plutôt que CSPA ou le spi calcul pour modéliser l'intrus nous permet de restreindre ce dernier à des manipulations de messages spécifiques. Comme nous l'avons noté précédemment, cet aspect est fort attrayant lorsque nous souhaitons formaliser des attaques de déni de service qui nécessitent peu de ressources.

Les formalisations de la confidentialité et de l'authentification présentées au Chapitre 6 généralisent celles introduites par Focardi & Gorrieri [44,48,50]. Cependant, notre approche basée sur l'interférence admissible, comparativement à celles simplement basées sur la non interférence, nous permet d'identifier des attaques inoffensives tôt dans les phases de conception du protocole. Par exemple, lors d'une analyse du protocole de Needham-Schröder, certaines « fausses attaques » détectées par les méthodes de Focardi & Gorrieri ne sont pas détectées par la notre. D'autre part, le concept d'action de déclassification nous évite d'augmenter notre langage de spécification à l'aide d'un système de déduction parallèle pour traiter les manipulations cryptographiques de messages. Dans CSPA, ces manipulations ne sont pas directement visibles de la sémantique des processus afin d'éviter de détecter l'interférence admissible qu'elles introduisent (par exemple, lors du cryptage d'un message). D'une part, SPPA permet une visualisation explicite des manipulations cryptographiques en terme d'actions et, d'autre part, notre formalisation de la confidentialité nous permet d'ignorer l'interférence admissible en identifiant les actions qui correspondent à des déclassifications.

Plus récemment, Bossi, Piazza & Rossi [22] ont proposé un modèle générique pour la formalisation de différentes propriétés de non interférence intransitive. Leur modèle, qui est élaboré dans le langage SPA, est une généralisation de celui présenté dans cette thèse. En considérant différentes relations d'équivalence observationnelle

sur les processus SPA, elles obtiennent ainsi une série de propriétés de flots d'information intransitifs vérifiables par *equivalence-checking*. En particulier, les auteures démontrent que notre propriété BNAI est équivalente à l'une de leurs propriétés (désignée par DP_BNDC).

Bien que notre méthode de validation contre les attaques de DoS est une interprétation dans SPPA du modèle de coût de Meadows [78], certaines distinctions persistes. Par exemple, nous ne considérons pas le coût cumulatif des comportements d'un intrus ou des autres participants au cours d'une exécution du protocole. Nous considérons plutôt le coût maximal des actions d'un comportement. Il s'avère qu'une extension du modèle proposé dans cette thèse qui permettrait de traiter le coût cumulatif des actions requiert une modification non triviale de la sémantique de l'opérateur d'observation $P/\mathcal{O}$ de SPPA. D'autre part, notre méthode s'appuie sur une représentation uniforme des capacités (mémoire et CPU) d'un intrus et d'une modélisation de celui-ci en terme de processus ennemi. Meadows opte plutôt pour une approche plus informelle qui exploite une relation de tolérance qui détermine combien de ressources le concepteur du protocole est prêt à dépenser afin d'assurer un certain niveau de sécurité.

Abadi, Blanchet & Fournet [2] ont analysé le protocole d'échange de clés *Just Fast Keying* (JFK) pour les communications IP sécuritaires. À partir d'une spécification dans le π-calcul, ils démontrent que le protocole JFK est robuste par rapport aux attaques de DoS. Plus spécifiquement, ils prouvent que le serveur (défendeur) n'effectuera des actions coûteuses seulement si l'intrus passe à travers les premières étapes du protocole. Leur formalisation de la robustesse face aux DoS est semblable à la notre car elle a pour but d'établir une dépendance causale entre les comportements de l'intrus et les actions coûteuses des participants. Cependant, leur validation s'appuie sur l'hypothèse qu'un intrus utilisant des adresses IP aléatoires ne sera pas capable d'intercepter les messages envoyées à cette adresse.

Cortier, Delaune & Lafourcade [33] offrent une synthèse des propriétés algébriques propres aux primitives cryptographiques. L'étude de ces propriétés est motivé par le fait qu'une analyse basée sur l'hypothèse classique de cryptage parfait,

dans laquelle le cryptage de données est traitée comme une boîte noire, ne parvient pas à détecter certaines attaques. Par exemple, le protocole peut nécessiter certaines propriétés algébriques ou une attaque sur un protocole peut dépendre de certaines propriétés des fonctions de cryptage, de signature et de hachage. Parmi les propriétés cryptographiques répertoriées par Cortier et al., nous comptons l'associativité, la commutativité, le OU exclusif et la propriété de préfixe. Nous sommes d'avis que ces propriétés sont définissables dans la logique pour les messages présentée à la Section 4.1 (directement au niveau des règles de satisfaction pour $\models a == b$). Ainsi, une implantation notre modèle symbolique pourrait prendre en considération ces propriétés, ce que très peu d'outils automatiques parviennent à faire. Parmi les outils qui le peuvent, notons le *Cryptographic Protocol Verifier* développé par Blanchet [18], dans lequel les propriétés sont définissables en termes de clauses de Horn. Le *NRL protocol analyzer* développé par Meadows [77] permet également une formalisation de ces propriétés en termes de règles de réduction.

La principale différence entre notre modèle symbolique et celui de Hennessy-Lin [57] réside dans le graphe de transitions symbolique : dans notre modèle, nous assignons à chaque état (processus) une formule qui offre une description précise des variables présentes dans le processus, tandis que le modèle de Hennessy-Lin nécessite une formule obtenue en considérant tous les chemins menant à l'état (processus). Notre modèle symbolique fut développé avec un objectif d'analyse de protocoles de sécurité en tête – il est donc essentiel d'avoir une description détaillée des valeurs symboliques à un moment précis du protocole afin d'augmenter l'efficacité de l'analyse. En effet, l'analyse de protocoles de sécurité requiert fréquemment une vérification des effets de l'introduction de valeurs aléatoires (par exemple, un message forgé envoyé par un intrus) sur certains participants du protocole. Dans ce contexte, notre notion de processus contraint nous permet d'identifier directement ces valeurs aléatoires qui pourraient mener à une attaque. Par exemple, l'analyse d'un protocole contre les attaques de DoS nécessite généralement de vérifier si un message forgé envoyé par un intrus peut engendrer l'exécution de fonctions coûteuses en terme de ressources (telles le décryptage de données ou la vérification

d'un message signé). Dans ce cas, une stratégie d'analyse basée sur les processus contraints consiste à vérifier, pour tout processus contraint obtenu à la suite d'une telle action coûteuse, les restrictions imposées par la formule sur la variable qui représente le message forgé : si tous les messages forgés satisfont la formule, alors nous pouvons conclure que le protocole est incapable de détecter et contrer une session frauduleuse ; si la formule est satisfaite seulement par un petit nombre de messages, alors nous pouvons conclure que le protocole est sécuritaire car la majorité des attaques initiées par un intrus auront été préalablement détectées. Une méthode similaire qui permet de détecter si un protocole est vulnérable face aux attaques de DoS fut proposée à la Section 6.5.

Boreale [20] a développé un modèle symbolique pour l'analyse de protocoles cryptographiques. Son modèle est basé sur une spécification du protocole dans le spi calcul sans les opérateurs de copie et de récursion. (Notons que notre modèle symbolique fonctionne avec les définitions récursives de SPPA.) Comme la majorité des modèles fondés sur le spi calcul, ce modèle suppose que le réseau de communication est entièrement contrôlé par l'environnement (c'est-à-dire l'intrus). Ainsi, un participant en attente d'un message peut, en principe, recevoir n'importe quel message que l'environnement est capable de produire. Par conséquent, les graphes de transitions associés à ces modèles sont souvent infinis (comme c'est le cas pour les algèbres de processus avec passage de paramètres). L'astuce générale derrière le modèle symbolique de Boreale est similaire à la notre : elle consiste à remplacer l'infinité de transitions provenant d'une action d'entrée par une seule transition symbolique dans laquelle le message reçu est représenté par une variable. Un état du protocole est ensuite représenté par un couple (s, P) formé d'un processus P et d'une trace symbolique s (c'est-à-dire une suite d'actions pouvant contenir des variables libres), dans laquelle est conservée l'historique des interactions entre le processus et l'environnement. Cette approche est donc similaire à celle de Hennessy-Lin. Par rapport à notre modèle symbolique, le concept de processus contraint nous permet de rassembler les couples (s, P) et (s', P) dès que les traces symboliques s et s' sont caractérisées par la même formule. Boreale démontre que la séman-

tique symbolique d'un couple (s, P) coïncide avec la sémantique conventionnelle du processus P. Cependant, aucune relation de bisimulation sur cette sémantique symbolique n'est offerte.

Fiore & Abadi [42] ont développé un modèle symbolique muni d'une sémantique de réduction symbolique qui s'inspire de celle de Boreale [20]. Plus spécifiquement, ils offrent une procédure qui, à partir d'une représentation finie d'un protocole évoluant dans un environnent hostile, détermine un ensemble de traces symboliques sur lesquelles des propriétés de sécurité peuvent être vérifiées. L'idée derrière leur approche consiste à réduire l'ensemble infini des traces d'un processus du spi calcul à un ensemble fini. Cette réduction est accomplie suite à une analyse symbolique des connaissances de l'environnement. Cette analyse algorithmique est requise car, contrairement à notre modèle symbolique, certaines traces symboliques provenant de leur sémantique symbolique ne correspondent pas à des traces du processus. Notons aussi que les modèles symboliques de Boreale et de Fiore-Abadi sont limités à des spécifications de la sécurité en terme de problème de traces (atteignabilité). Ils s'intéressent tout particulièrement à des propriétés qui sont de la forme « dans toute exécution du protocole, l'action α se produit avant l'action β ».

Echahed & Prost [39] ont proposé une méthode d'analyse de la confidentialité de l'information pour certaines applications basées sur l'utilisation de mots de passe (guichet automatique bancaire, procédure de *login*, etc.), pour lesquels une analyse de non interférence est souvent insuffisante. Ils introduisent un concept de *fonction de conversion* qui permet de modifier le niveau de confidentialité des messages, et une notion d'*interférence contrôlée* qui s'assure que toute interférence entre les divers niveaux de confidentialité provient d'une fonction de conversion pour laquelle une déclassification de l'information est autorisée. Echahed & Prost donnent une définition formelle et un algorithme de vérification pour l'interférence contrôlée dans le contexte de la programmation déclarative concurrente. Cette méthode de validation s'appuie sur des concepts analogues à ceux qui ont inspirée notre méthode basée sur la propriété BNAI. Une combinaison de ces deux approches devrait être parmi nos objectifs de recherche immédiats. L'interférence contrôlée nous

procure un algorithme efficace par vérification de typage qui pourrait s'avérer plus précis que BNAI dans le sens que l'interférence admissible est permise jusqu'à la dernière action de déclassification, après quoi tout flot d'information qui cause de l'interférence est interdit. Par exemple, après une déclassification, formalisée par l'utilisation d'une certaine fonction de conversion, il pourrait y avoir de l'interférence entre deux niveaux de confidentialité provenant indirectement de l'utilisation de la fonction de conversion. Ce type d'interférence est autorisé selon leur approche.

L'approche présentée ci-dessus est similaire à celle de Myers, Sabelfeld & Zdancewic [85] qui utilise une propriété de robustesse afin de caractériser les programmes pour lesquels les mécanismes de déclassification ne peuvent pas être exploités par un intrus afin de provoquer des divulgations d'information non souhaitées. Cette propriété de *déclassification robuste* est vérifiée à partir d'une analyse du typage du programme qui garantit qu'aucun intrus ne peut contrôler quelles données seront déclassifiées. Cependant, notons que le langage de spécification utilisé par Myers et al. ne considère pas la concurrence. Comme c'est le cas pour notre modèle, mais pas pour celui de Echahed-Prost, les déclassifications sont directement déclarées au niveau des actions. Par conséquent, il est possible de restreindre les déclassifications à certaines portions du treillis contenant les divers niveaux de sécurité. D'autre part, leur propriété de déclassification robuste est définie selon une approche analogue à celle de Boreale et al. [21], c'est-à-dire en exigeant que les programmes $P[a]$ et $P[a']$ soient indistinguables pour tous tuplets de messages a et a' provenant de l'intrus.

8.3 Nouvelles voies de recherche

Un outil d'analyse de protocoles de sécurité basé sur l'algèbre de processus SPPA et la propriété d'interférence admissible a été développé au *Laboratoire CRAC* de l'*École Polytechnique de Montréal*. Cet outil, nommé ASPiC, prend comme entrée un protocole spécifié dans une extension symbolique de SPPA appelée *Security Process Algebra for Symbolic Manipulations* (SPASM) [14]. Cet analyseur de protocoles a été utilisé avec succès pour vérifier certaines propriétés propres aux

protocoles cryptographiques et aux protocoles de commerce électronique. Plus spécifiquement, ASPiC implante la méthode de vérification par bisimulation de BNAI et les propriétés vérifiées par celui-ci, dont la confidentialité, l'authentification, la non répudiation et l'équité, sont directement exprimées en terme d'interférence admissible.

Toujours au *Laboratoire CRAC*, nous développons présentement un outil qui génère la spécification en SPPA d'un protocole à partir de sa spécification dans le langage *High Level Protocol Specification Language* (HLPSL) [13]. Le langage HLPSL est un langage de haut niveau (un peu plus détaillé que la notation Alice & Bob) développé dans le cadre du projet Européen AVISPA (`www.avispa-project.org`). Cet outil de traduction nous permettra d'utiliser un grand nombre de protocoles déjà spécifiés en HLPSL et de valider ceux-ci selon les diverses méthodes présentées dans cette thèse. En fait, une fois un protocole traduit en SPPA, plusieurs options s'offrent à nous afin d'en obtenir une analyse. Tout d'abord, nous pourrons utiliser l'outil présenté ci-dessus pour analyser le protocole, après avoir généré la spécification SPASM du protocole à partir de celle en SPPA. D'autre part, nous pourrons effectuer une vérification du protocole directement à partir de la sémantique avec passage de paramètres de SPPA. Finalement, nous pourrons utiliser le modèle symbolique présenté au Chapitre 4. Comme nous l'avons constaté à la Section 4.3.1, une spécification du protocole en terme de processus contraint s'obtient facilement de sa spécification SPPA. L'outil de traduction pourrait générer une spécification symbolique en SPPA, directement à partir de sa spécification HLPSL, qui formalise les connaissances initiales des participants à l'aide d'une formule jointe au processus contraint correspondant au protocole. Une vérification de ce processus contraint est ensuite accomplie selon l'une des applications de BNAI présentée dans cette thèse. Des méthodes spécialement adaptées à notre modèle symbolique pour la vérification de propriétés de confidentialité, d'authentification et de robustesse contre le DoS sont présentées à la Section 6.4.

Une autre ramification à notre recherche consiste à développer des propriétés de sécurité, tout particulièrement pour les protocoles de commerce électronique, à par-

tir d'interprétations de la propriété BNAI. Cette approche nous permet de définir des propriétés de sécurité à l'aide des flots d'information et de profiter des caractérisations algébriques de BNAI dans le contexte d'une méthode de preuve pour leur vérification. Une spécification de la propriété d'anonymat a été obtenue à partir de BNAI par Brlek, Hamadou & Mullins [25]. D'autre part, nous sommes d'avis qu'une propriété de non répudiation s'obtient facilement à partir d'une interprétation du concept de non interférence. En effet, il semblerait que la non répudiation soit reliée à la satisfaction de l'interférence plutôt que de la non interférence, c'est-à-dire la non répudiation se ramène à vérifier qu'un participant a causé de l'interférence sur un certain message afin de s'assurer que ce message est effectivement une conséquence directe du comportement du participant. Si nous parvenons à démontrer que l'occurrence de ce message est une conséquence d'une certaine action du participant, celui-ci ne peut pas nier son lien avec le message. Cependant, nous ne savons pas encore comment interpréter le concept d'admissibilité et d'attaque admissible dans ce contexte de la non répudiation.

Focardi, Gorrieri & Martinelli [49] ont récemment proposé une extension temporelle de SPA, de même qu'une relation de bisimulation faible temporelle sur celle-ci. À partir de ce modèle, ils étendent dans un contexte temporel certaines propriétés de non interférence, dont SNNI et BSNNI. Due à la simplicité de leur approche, nous sommes d'avis qu'une extension semblable à notre algèbre de processus SPPA s'obtient facilement. Du coup, nous pourrions en déduire une extension temporelle de notre propriété BNAI. Cependant, l'extension temporelle de l'opérateur d'observation $P/\mathcal{O}$ de SPPA pourrait s'avérer assez complexe car cet opérateur est un raffinement non trivial de l'opérateur de masquage de SPA et l'extension temporelle de celui-ci doit respecter une propriété de progression maximale (c'est-à-dire, les actions τ doivent avoir préséances sur l'écoulement du temps).

Une interprétation de la non interférence dans le contexte du π-calcul est proposée par Hennessy [56,59]. En s'inspirant de ces travaux, il serait intéressant de formaliser la notion d'admissibilité dans cette algèbre de processus. Une telle interprétation passerait par une extension de SPPA afin d'y intégrer les concepts propres

au π-calcul, tout particulièrement la mobilité des processus. Cette extension nécessite une généralisation de notre algorithme de $\mathcal{O}$-bisimulation (Définition 3.19) au π-calcul. Une approche basée sur les flots d'information pour le π-calcul fut récemment proposée par Bodei et al. [19], et pour laquelle les auteurs présentent une application à l'analyse du contrôle de flots d'information dans les protocoles cryptographiques. Cependant, nous conjecturons que les propriétés de sécurité établies par Bodei et al., plus précisément les propriétés d'« *aucune divulgation* » (*no leaks*) et « *pas de lecture vers le haut/pas d'écriture vers le bas* » (*no read-up/no write-down*), ne permettent pas la détection de canaux clandestins dans les protocoles cryptographiques, contrairement aux propriétés de flots d'information telles la non interférence et l'interférence admissible.

Au niveau de notre modèle symbolique, nous avons comme projet d'élaborer un algorithme pour déterminer si deux formules sont équivalentes. La comparaison de formules est fréquemment utilisée dans la construction de la sémantique symbolique d'un processus contraint. Un tel algorithme devrait se déduire de l'algorithme de décision pour notre logique de messages complétée avec les opérateurs logiques habituels, dont un quantificateur universel.

Comme nous l'avons mentionné au Chapitre 6, le concept de processus contraint introduit dans notre modèle symbolique est tout à fait propice à l'analyse de la vulnérabilité au DoS. Cette extension de notre méthode d'analyse de la robustesse face au DoS nous permettrait d'obtenir un aperçu immédiat des valeurs symboliques présentes à un moment précis du protocole. En particulier, cette approche nous permettrait de déterminer quels messages forgés par un intrus persisteront jusqu'à la fin du protocole : si la plupart de ces messages se rendent jusqu'à la fin, alors nous pourrons conclure que le protocole n'est pas sécuritaire ; si seulement un petit nombre de ces messages se rendent jusqu'à la fin, alors nous pourrons conclure que la majorité des attaques furent contrées. Nous prévoyons développer une seconde extension à notre méthode d'analyse de DoS qui pourra prendre en compte le coût cumulatif des ressources. Cette approche pourrait ainsi formaliser l'allocation et la désallocation de ressources mémoire d'un participant lors d'un protocole.

Nous sommes présentement à l'élaboration d'un modèle dans lequel les quantités de ressources disponibles pour chaque participant à un protocole sont exprimées conjointement avec le processus SPPA qui décrit le déroulement du protocole. Nous prévoyons modéliser les dépenses de ressources à l'aide d'un simple vecteur de ressources, d'un semi-anneau, ou d'une autre structure mathématique pour laquelle un algorithme efficace est connu pour déterminer un état critique du protocole, c'est-à-dire un moment dans le déroulement du protocole où il y a épuisement de ressources. À partir de ce modèle pour les ressources, nous pouvons facilement élaborer une sémantique pour un processus SPPA P (et éventuellement une sémantique symbolique pour un processus contraint) combiné à une structure de ressources T. Cette sémantique aurait la forme suivante :

$$\frac{P \xrightarrow{\alpha} P'}{\langle P,\ T\rangle \xrightarrow{\alpha} \langle P',\ T + \rho(\alpha)\rangle}$$

où $T + \rho(\alpha)$ désigne la structure de ressources obtenue de T suite à l'exécution de l'action α (où $\rho(\alpha)$ qui désigne le coût de α). À partir de cette sémantique, il suffirait d'utiliser un algorithme propre à la structure de ressources afin de déterminer l'état du protocole pour lequel le niveau de ressources est le plus bas (ou déterminer si le déroulement du protocole peut mener à une situation de *overflow* ou de *underflow*). Notons que ce genre de sémantique est tout à fait propice à une analyse du protocole qui prend en compte le coût cumulatif des ressources. De plus, si l'interprétation de la vulnérabilité au DoS se ramène à déterminer si un certain état critique est atteignable de l'état initial du protocole, alors notre méthode pourrait s'appuyer sur une formulation à l'aide d'une formule du μ-calcul de la propriété de robustesse face au DoS.

D'autre part, nous travaillons présentement au *Laboratoire CRAC* sur l'élaboration d'un modèle stochastique pour l'analyse des attaques de DoS. L'idée derrière ce modèle consiste à développer un langage de spécification pour les protocoles de sécurité qui combine les algèbres de processus, plus spécifiquement SPPA, et les chaînes de Markov. Ce modèle de spécification pourrait s'inspirer du modèle

d'analyse de performance de systèmes informatiques proposé par Brinksma & Hermanns [24].

BIBLIOGRAPHIE

[1] M. Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5) :749–786, Septembre 1999.

[2] M. Abadi, B. Blanchet, and C. Fournet. Just fast keying in the pi calculus, 2003.

[3] M. Abadi, M. Burrows, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1) :18–36, 1990.

[4] M. Abadi and A. D. Gordon. Reasoning about cryptographic protocols in the spi calculus. In *CONCUR'97 : Concurrency Theory*, volume 1243, pages 59–73, Berlin, Germany, 1997. Springer-Verlag.

[5] M. Abadi and A. D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4) :267–303, 1998.

[6] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols : The spi calculus. *Information and Computation*, 148(1) :1–70, Janvier 1999.

[7] M. Abadi and L. Lamport. Conjoining specifications. *ACM TOPLAS*, 17(3) :507–534, Mai 1995.

[8] P. A. Abdulla, B. Jonsson, and A. Nylén. Modelling and automated verification of authentication protocols. In *Proc. of Dimacs workshop on design and formal verification of security protocols*, 1997.

[9] R.M. Amadio and W. Charatonik. On name generation and set-based analysis in the dolev-yao model. In *Procedings of CONCUR 02*. Springer-Verlag, 2002.

[10] R.M. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. Technical Report 3915, INRIA, Mars 2000.

[11] R.M. Amadio and C. Meyssonnier. On decidability of the control reachability problem in the asynchronous pi-calculus. *Nordic Journal of Computing*, 9(2) :70–101, 2002.

[12] A. Arnold. *Finite transition systems and semantics of communicating sytems*. Prentice-Hal, 1994.

[13] AVISPA. Deliverable d2.1 : The high level protocol specification language. Technical Report IST-2001-39252, Information Society Technologies, Août 2003.

[14] G. Bastien. Aspic : un outil d'analyse symbolique automatisée de protocoles cryptographiques basé sur le modèle de flus d'information. Master's thesis, École Polytechnique de Montréal, Montréal, Québec, Septembre 2004.

[15] D.E. Bell and L.J. LaPadula. Secure computer system : Unified exposition and multics interpretation. Technical Report MTR-2997, Mitre Corp., Bedford, Mass., USA, Juin 1976.

[16] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, E. Van Herreweghen, and M. Waidner. Design, implementation and deployment of the iKP secure electronic payment system. *IEEE Journal of Selected Areas in Communications*, 18(4) :611–627, Avril 2000.

[17] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, and M. Waidner. ikp – a family of secure electronic payment protocols. In *First USENIX Workshop on Electronic Commerce*, Mai 1995.

[18] B. Blanchet. Cryptographic protocol verifier user manual, 2004.

[19] C. Bodei, P. Degano, F. Nielson, and H. Nielson. Static analysis for the π-calculus with applications to security. *Information and Computation*, 168 :68–92, 2001.

[20] M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proceedings of ICALP'01*, 2001.

[21] M. Boreale, R. De Nicola, and R. Pugliese. Proof techniques for cryptographic processes. In *Logic in Computer Science*, pages 157–166, 1999.

[22] A. Bossi, C. Piazza, and S. Rossi. Modelling downgrading in information flow security. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, pages 187–201. IEEE Computer Society Press, 2004.

[23] G. Boudol. Notes on algebraic calculi of processes. In *Logic and Models of Concurrent Systems*, NATO ASI Series F-13, pages 261–303. Springer, 1985.

[24] E. Brinksma and H. Hermanns. Process algebra and markov chains. In *Lectures on Formal Methods and Performance Analysis : First EEF/Euro Summer School on Trends in Computer Science*, number 2090 in LNCS, pages 183–231. Springer-Verlag, 2002.

[25] S. Brlek, S. Hamadou, and J. Mullins. Aset, un protocole anonyme et sécuritaire pour les transactions éléctroniques. In *Actes du 2ème rencontre francophone sur Sécurité et Architecture Réseaux*, Nancy, France, 2003.

[26] P. Broadfoot and G. Lowe. Analysing a stream authentication protocol using model checking. In *Proceedings of the 7th European Symposium on Research in Computer Security (ESORICS)*, 2002.

[27] M. Bugliesi, A. Ceccato, and S. Rossi. Non-interference proof techniques for the analysis of cryptographic protocols, 2002.

[28] L. Cardelli, G. Ghelli, and A. D. Gordon. Types for the ambient calculus. *Information and Computation*, 177 :160–194, 2002.

[29] L. Cardelli and A. D. Gordon. Mobile ambients. Available at www.cs.york.ac.uk/jac/papers/.

Theoretical Computer Science, 240 :177–213, 2000.

[30] J. Clark and M. Jacob. A survey of authentication protocol literature : Version 1.0, Novembre 1997.

[31] E. M. Clarke and E. A. Emerson. Design and verification of synchronizing skeletons using branching time temporal logic. In *Logics of Programs Workshop*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.

[32] V. Cortier. Observational equivalence and trace equivalence in an extension of spi-calculus. application to cryptographic protocols analysis. Technical Report LSV-02-3, Lab. Specification and Verification, ENS de Cachan, Cachan, France, Mars 2002.

[33] Véronique Cortier, Stéphanie Delaune, and Pascal Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 2005. To appear.

[34] P.J. Criscuolo. Distributed denial of service trin00, tribe flood network, tribe flood network 2000, and stacheldraht. Technical Report CIAC-2319, Lawrence Livermore National Laboratory, Février 2000.

[35] F. Cuppens and C. Saurel. Towards a formalization of availability and denial of ssrvice. In *Information Systems Technology Panel Symposium on Protecting Nato Information Systems in the 21st century*, Washington, 1999.

[36] S. Dietrich, N. Long, and D. Dittrich. Analyzing distributed denial of service tools : The shaft case. In *Proceedings of USENIX LISA*, 2000.

[37] D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29(2) :198–208, 1983.

[38] A. Durante, R. Focardi, and R. Gorrieri. CVS : A compiler for the analysis of cryptographic protocols. In *Proceedings of 12th IEEE Computer Security Foundations Workshop*, pages 203–212. IEEE Computer Society, Juin 1999.

[39] R. Echahed and F. Prost. Controlling interferences, 2004. To appear.

[40] T. Fine. Constructively using noninterference to analyze systems. In *IEEE Symposium on Security and Privacy*, pages 162–169, 1990.

[41] T. Fine. Defining noninterference in the temporal logic of actions. In *IEEE Symposium on Security and Privacy*, pages 12–21, 1996.

[42] M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages pp. 160–173, 2001.

[43] R. Focardi. Using entity locations for the analysis of authentication protocols. In *Proceedings of Sixth Italian Conference on Theoretical Computer Science (ICTCS'98)*, Novembre 1998.

[44] R. Focardi, A. Ghelli, and R. Gorrieri. Using non interference for the analysis of security protocols. In H. Orman and C. Meadows, editors, *Proceedings*

of the DIMACS Workshop on Design and Formal Verification of Security Protocols, Rutgers University, Septembre 1997.

[45] R. Focardi and R. Gorrieri. A classification of security properties for process algebras. *Journal of Computer Security*, 3(1) :5–33, 1994/1995.

[46] R. Focardi, R. Gorrieri, and F. Martinelli. Information flow analysis in a discrete-time process algebra. In *PCSFW : Proceedings of The 13th Computer Security Foundations Workshop*. IEEE Computer Society Press, 2000.

[47] R. Focardi, R. Gorrieri, and F. Martinelli. Non interference for the analysis of cryptographic protocols. In *Proceedings of ICALP'00*, Geneva (Switzerland), Juillet 2000.

[48] R. Focardi, R. Gorrieri, and F. Martinelli. Secrecy in security protocols as non interference. In S. Schneider and P. Ryan, editors, *Proceedings of DERA/RHUL Workshop on Secure Architectures and Information Flow*, volume 32. Elsevier ENTCS, 2000.

[49] R. Focardi, R. Gorrieri, and F. Martinelli. Real-time information flow analysis. *IEEE Journal on Selected Areas in Communications*, 21(1) :20–35, Janvier 2003.

[50] R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *Proceedings of World Congress on Formal Methods (FM'99)*, volume 1708 of *LNCS*, pages 794–813. Springer, 1999.

[51] G. Ferrari, G. Ferro, S. Gnesi, U. Montanari, M. Pistore, and G. Ristori. An automata based verification environment for mobile processes. In E. Brinksma, editor, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 275–290, Enschede, The Netherlands, 1997. Springer Verlag, LNCS 1217.

[52] J.A. Goguen and J. Meseguer. Security policies and security models. In *Proceedings 1982 IEEE Symposium on Research in Security and Privacy*, pages 11–20, Avril 1982.

[53] J.A. Goguen and J. Meseguer. Unwinding and inference control. In *Proceedings of the IEEE Symp. on Research in Security and Privacy*, pages 75–285, Oakland, CA, 1984. IEEE Computer Society.

[54] L. Gong and P. Syverson. Fail-stop protocols : An approach to designing secure protocols. In *Dependable Computing for Critical Applications*, volume 5, pages 79–100. IEEE Computer Society, 1998.

[55] J.T. Haigh and W.D. Young. Extending the noninterference version of mls for sat. *IEEE Trans. on Software Engineering*, 13(2) :141–150, 1987.

[56] M. Hennessy. The security picaculus and non-interference. Technical Report 05/2000, School of Cognitive and Computing Sciences, University of Sussex, UK, Brighton BN1 9QH, Novembre 2000.

[57] M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138 :353–389, 1995.

[58] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1) :137–161, 1985.

[59] M. Hennessy and J. Riely. Information flow vs resource access in the asynchronous pi-calculus. *ACM Transactions on Programming Languages and Systems*, 5 :556–591, 2002.

[60] Matthew Hennessy and James Riely. Type-safe execution of mobile agents in anonymous networks. In Jan Vitek and Christian Damsgaard Jensen, editors, *Proceedings of MOS '98*, volume 1603 of *LNCS*. Springer, 1999. Full version as CogSci Report 3/98, University of Sussex, Brighton.

[61] C.A.R. Hoare. *Communicating sequential processes*. Prentice-Hall, 1985.

[62] IEEE Computer Society. *1995 IEEE Symp. on Research in Security and Privacy*, Oakland, CA, Mai 1995.

[63] ISO. Information technology - security techniques - entity authentication mechanisms ; part 1 : General model. ISO/IEC 9798-1, second edition, Septembre 1991.

[64] D. Janin and I. Walukiewicz. Automata for the mu-calculus and related results. In *MFCS'95*, volume 969 of *Lecture Notes in Computer Science*, pages 552–562. Springer-Verlag, 1995.

[65] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *CONCUR '96 : Concurrency Theory, 7th International Conference*, volume 1119, pages 263–277, Pisa, Italy, 1996. Springer-Verlag.

[66] Steve Kremer and Mark D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In Mooly Sagiv, editor, *Programming Languages and Systems – Proceedings of the 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, Edinburgh, U.K., Avril 2005. Springer.

[67] P. Krishnan. Architectural ccs. *Formal Aspects of Computing : The International Journal of Formal Methods*, 8(2) :162–187, 1996.

[68] S. Lafrance and J. Mullins. A generic enemy process for the analysis of cryptographic protocols. In C.Rattray and M. Sveda, editors, *Proceedings of the Joint Workshop on Formal Specifications of Computer-Based Systems - IEEE ECBS'2002-IFIP WG10.1*, pages 1–9, Lund Sweden, 2002.

[69] S. Lafrance and J. Mullins. An information flow method to detect denial of service vulnerabilities. In V. Dvorak, M. Sveda, C. Rattray, and J. W. Rozenblit, editors, *Formal Specifications of Computer-Based Systems*, volume 9, pages 1258–1260. Journal of Universal Computer Science, Novembre 2003.

[70] S. Lafrance and J. Mullins. Symbolic approach to the analysis of security protocols. In Iliano Cervesato, editor, *Proceedings of Foundations of Computer Security*, 2003.

[71] S. Lafrance and J. Mullins. Using admissible interference to detect denial of service vulnerabilities. In J. M. Morris, B. Aziz, and F. Oehl, editors,

Sixth International Workshop in Formal Methods. Electronic Workshops in Computing (eWiC) by British Computer Society (BCS), 2003.

[72] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3) :872–923, Mai 1994.

[73] L. Lamport and M. Abadi. Preserving liveness : Comments on 'safety and liveness from a methodological point of view'. *Information Processing Letters*, 40(3) :141–142, Novembre 1991.

[74] G. Lowe. Breaking and fixing the needham-schröder public-key protocol using FDR. In *Proceedings of TACAS'96*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, 1996.

[75] G. Lowe. Quantifying information flow. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, 2002.

[76] J. McLean. Security models and information flow. In *Proceedings of the IEEE Symp. on Research in Security and Privacy*, pages 180–187, Oakland, CA, May 1990. IEEE Computer Society.

[77] C. Meadows. The NRL protocol analyzer : An overview. *Journal of Logic Programming*, 26(2) :113–131, 1996.

[78] C. Meadows. A cost-based framework for analysis of denial of service networks. *Journal of Computer Security*, 9(1/2) :143–164, 2001.

[79] C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *Proceedings of DISCEX 2000*, pages 237–250. IEEE Computer Society Press, 2003.

[80] J. Millen. A resource allocation model for denial of service. In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, pages 137– 147. IEEE Computer Society Press, 1992.

[81] R. Milner. *Communication and concurrency*. Prentice-Hall, 1989.

[82] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. *Information and Computation*, 100(1) :1–77, Septembre 1992.

[83] J. Mullins. Nondeterministic admissible interference. *Journal of Universal Computer Science*, 6(11) :1054–1070, 2000.

[84] J. Mullins and S Lafrance. Bisimulation-based non-deterministic admissible interference with applications to the analysis of cryptographic protocols. *International Journal in Information and Software Technology*, pages 1–25, 2002.

[85] A.C. Myers, A. Sabelfeld, and S. Zdancewic. Enforcing robust declassification. In *CSFW '04 : Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, page 172, Washington, DC, USA, 2004. IEEE Computer Society.

[86] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12) :993–999, 1978.

[87] D. Park. Concurrency and automata on infinite sequences. In *5th GIConference on Theorical Computer Science*, volume 104 of *LNCS*. Springer-Verlag, 1981.

[88] V. Paxson. An analysis of using reflectors in distributed denial-of-service attacks, 2001.

[89] A. Perrig, R. Canettiand, J.D. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. In *Proc. of IEEE Security and Privacy Symposium*, pages 56–73, Oakland, CA, May 2000.

[90] S. Pinsky. Absorbing covers and intransitive non-interference. In *Proceedings of the IEEE Symp. on Research in Security and Privacy* [62], pages 102–113.

[91] J. Rathke. *Symbolic Techniques for Value-passing Calculi*. PhD thesis, University of Sussex, Brighton, UK, March 1997.

[92] G. Rohrmair and G. Lowe. Using CSP to detect insertion and evasion possibilities within the intrusion detection area. In *Proceedings of BCS Workshop on Formal Aspects of Security*, 2002.

[93] A.W. Roscoe. Modelling and verifying key-exchange using CSP and FDR. In *8th Computer Security Foundation Workshop*. IEEE Computer Society Press, 1995.

[94] A.W. Roscoe and M.H. Goldsmith. What is intransitive noninterference. In *12th IEEE Computer Security Foundations Workshop*, 1999.

[95] J. Rushby. Noninterference, transitivity and channel-control security policies. Technical Report CSL-92-02, SRI International, Menlo Park CA, USA, December 1992.

[96] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is np-complete. In *14th IEEE Computer Security Foundations Workshop*, pages 174–190, 2001.

[97] F. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 2(4), March 2000.

[98] S. Schneider. Modelling security properties with CSP. Technical Report CSD-TR-96-04, Royal Holloway, University of London, 1996.

[99] S. Schneider. Security properties and CSP. In *IEEE Symposium on Security and Privacy*, pages 174–187, 1996.

[100] S. Schneider. Verifying authentication protocols in CSP. *IEEE Transactions of Software Engineering*, 24(8) :447–479, 1998.

[101] C.L. Schuba, I.V. Krsul, M.G. Kuhn, E.H. Spafford, A. Sundaram, and D. Zamboni. Analysis of a denial of service attack on TCP. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 208–223. IEEE Computer Society Press, Mai 1997.

[102] C. Stirling. Games and modal mu-calculus. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1055 of *Lecture Notes in Computer Science*, pages 298–312. Springer-Verlag, 1996.

[103] D. Stoller. A bound on attacks on payment protocols. In *Logic in Computer Science*, pages 61–70, 2001.

[104] R. Streett and E. Emerson. An automata theoretic procedure for the propositional mu-calculus. *Information and Computation*, 81 :249–264, 1989.

[105] D. Sutherland. A model of information. In *Proceedings of the IEEE Symp. on Research in Security and Privacy*, pages 11–20, Oakland, CA, 1986. IEEE Computer Society.

[106] I. Sutherland. Relating bell-lapadula-style security models to information models. In *Proceedings of the Computer Security Foundations Workshop*, pages 112–126, Franconia, NH, 1988.

[107] R. van Glabbeek. Notes on the methodology of CCS and CSP. *Theoretical Computer Science*, 177 :329–349, 1997.

[108] T. Woo and S. Lam. Authentication for distributed systems. *IEEE Computer*, 25(1) :39–52, Janvier 1992.

[109] T.Y.C. Woo and S.S. Lam. A lesson in authentication protocol design. *ACM Operating Systems Review*, 28(3), 1994.

[110] C. Yu and V. D. Gligor. A formal specification and verification method for the prevention of denial of service. In *1988 IEEE Symposium on Security and Privacy*, volume 117, pages 187–202. IEEE Computer Society Press, Avril 1988.

[111] S. Zdancewic. A type system for robust declassification. In *Proceedings of the Nineteenth Conference on the Mathematical Foundations of Programming Semantics*. Electronic Notes in Theoretical Computer Science, Mars 2003.

[112] S. Zdancewic and A.C. Myers. Robust declassification. In *Proceedings of 14th IEEE Computer Security Foundations Workshop*, pages 15–23, Cape Breton, Nova Scotia, Canada, Juin 2001.