Sécurité des protocoles cryptographiques : aspects logiques et calculatoires

Mathieu Baudet

THÈSE

présentée à l'École Normale Supérieure de Cachan

pour obtenir le grade de

Docteur de l'École Normale Supérieure de Cachan

par : Mathieu Baudet
Spécialité : Informatique

Sécurité des protocoles cryptographiques : aspects logiques et calculatoires

Thèse soutenue le 16 janvier 2007

Composition du jury :

– Martín Abadi	examinateur
– Bruno Blanchet	examinateur
– Jean Goubault-Larrecq	directeur de thèse
– Ralf Küsters	rapporteur
– Yassine Lakhnech	rapporteur
– Jacques Stern	président du jury

Laboratoire Spécification et Vérification (LSV), INRIA Futurs, ENS Cachan, CNRS
61, avenue du Président Wilson, 94235 Cachan Cedex

Généré par LaTeX le 24 janvier 2007.

Résumé

Cette thèse est consacrée au problème de la vérification automatique des protocoles cryptographiques d'un point de vue logique et calculatoire.

Dans une première partie, nous abordons la sécurité des protocoles dans le cadre logique (« formel »). Nous montrons tout d'abord comment spécifier, dans un langage de processus proche du pi-calcul appliqué, différentes propriétés de sécurité des protocoles, en particulier : le secret simple, l'authentification et la résistance aux attaques par dictionnaire. L'originalité de ce langage réside dans l'existence d'une sémantique symbolique valable pour toute théorie équationnelle, c'est-à-dire quelles que soient les primitives cryptographiques modélisées. Dans le cas d'un nombre borné de sessions, grâce à cette sémantique symbolique, nous obtenons ensuite la décidabilité dans (co-)NP de toutes les propriétés de sécurité énoncées ci-dessus, pour une classe générale de théories équationnelles dites *sous-terme-convergentes*.

La seconde partie traite de la justification cryptographique des modèles logiques. Il s'agit d'étudier dans quelle mesure l'absence d'attaques logiques sur un protocole implique l'existence d'une preuve de sécurité dans le modèle cryptographique usuel (dit *modèle calculatoire*). Nous nous intéressons ici plus particulièrement à une notion formelle d'indistinguabilité héritée du pi-calcul appliqué : l'*équivalence statique*. Étant donnée une théorie équationnelle E modélisant les propriétés algébriques des primitives cryptographiques, deux messages sont statiquement équivalents, intuitivement, si et seulement s'ils vérifient les mêmes équations (modulo E) du point de vue de l'attaquant. Dans un premier temps, nous étudions le lien entre l'équivalence statique et l'indistinguabilité cryptographique usuelle pour une théorie E quelconque. Nous utilisons ensuite les techniques développées pour montrer la sûreté cryptographique de l'équivalence statique pour un ensemble de primitives réalistes comprenant trois formes de chiffrement, ainsi que des données vulnérables aux attaques par dictionnaire. Ainsi, dans les conditions de ce théorème, toute preuve (logique) d'équivalence statique implique l'indistinguabilité (cryptographique) des données modélisées.

Mots clés

Protocoles cryptographiques, lien entre méthodes formelles et cryptographie, théories équationnelles, algèbres de processus.

Abstract

This thesis is dedicated to the automatic verification of cryptographic protocols in the logical and computational settings.

The first part concerns the security of procotols in the logical ("formal") framework. To begin with, we show how to specify various security properties in a concurrent language close to the applied pi-calculus. These include notably: simple secrecy, authentication, and resistance to dictionary attacks. Our language differs from existing work in so far as it is equipped with a symbolic semantics suitable for any equational theory, that is, whatever cryptographic primitives are modeled. In the case of a bounded number of sessions, thanks to this symbolic semantics, we prove the (co-)NP-decidability of every security property in the list above, for a general class of equational theories called subterm convergent.

The second part deals with the computational soundness of logical models. The main question here is to what extent the fact that no logical attack exists on a protocol implies that it is provably secure in the usual cryptographic model (called the *computational model*). We concentrate on static equivalence, a formal notion of indistinguishability originating from the applied pi-calculus. Given an equational theory E representing the algebraic properties of the primitives, static equivalence asks, intuitively, whether an attacker can distinguish two messages by exhibiting an equation which holds (modulo E) on one message but not on the other. First, we study the relationship between static equivalence and the usual cryptographic indistinguishability for any theory E. Then, we use the same techniques to show the computational soundness of static equivalence for a significant set of primitives, including three kinds of encryption as well as data vulnerable to dictionary attacks. Therefore, under the conditions of this theorem, any (logical) proof of static equivalence between two messages implies their (computational) indistinguishability.

Keywords

Cryptographic protocols, computational soundness of formal methods, equational theories, process algebras.

Remerciements

Je tiens à remercier en premier lieu les membres de mon jury.

Jacques Stern a bien voulu en être le président ; je l'en remercie chaleureusement.

Je tiens également à remercier particulièrement Ralf Küsters et Yassine Lakhnech d'avoir accepté la lourde tâche d'être rapporteurs.

Je suis très reconnaissant à Martín Abadi et à Bruno Blanchet de m'avoir fait l'amitié et l'honneur de participer à mon jury, ainsi que pour leurs nombreux commentaires sur le manuscrit.

Je tiens enfin à remercier Jean Goubault-Larrecq pour ses encouragements et ses conseils précieux tout au long de ma thèse, ainsi que pour le temps accordé à la relecture de ce document.

Cette thèse doit beaucoup aux collaborations et aux discussions fructueuses que j'ai pu entretenir avec Véronique Cortier, Martín Abadi, Steve Kremer, Bogdan Warinschi, Cédric Fournet, Bruno Blanchet, David Lubicz, Yassine Lakhnech, Laurent Mazaré, Stéphanie Delaune, Florent Jacquemard, et plus généralement les membres du projet SECSI. Je les en remercie vivement.

Mon travail a beaucoup bénéficié de l'atmosphère sympathique et studieuse du LSV, dont je remercie amicalement tous les membres. Je tiens également à remercier chaleureusement mes collègues de la DCSSI pour leur patience et leur compréhension pendant la (difficile !) phase finale de rédaction de ce manuscrit.

Enfin, je remercie ma famille et mes amis pour leur aide et leur soutien précieux durant ces années bien remplies.

Table des matières

Chapitre 1

Introduction

Lorsque l'on communique à distance, il n'est pas possible de voir ou de parler directement à son interlocuteur. Dans ces conditions, comment être sûr
- que l'on parle à la bonne personne (*authenticité*),
- que nos propos ne sont pas modifiés (*intégrité*), ou encore
- que la conversation n'est pas espionnée (*confidentialité*) ?

Si ces trois questions — entre autres — se posent depuis longtemps aux gouvernements et aux généraux, il est remarquable qu'avec l'essor des nouvelles technologies, le grand public soit concerné à son tour.

Le réseau Internet, tout d'abord, s'est développé considérablement ces dernières décennies, au point de conditionner désormais une part importante de l'activité économique. Conçu à l'origine comme un réseau entre différents organismes de confiance[1], Internet est vulnérable à de nombreuses attaques — une technique célèbre étant par exemple l'usurpation d'adresse IP (*IP spoofing* [Wik]). En outre, il est bien connu que le courrier électronique usuel, via Internet, ne donne aucune garantie d'authenticité, d'intégrité ou de confidentialité des messages aux utilisateurs.

La téléphonie mobile, ensuite, a pris en quelques années une place incontournable dans la société. Or, par définition, celle-ci repose sur l'utilisation d'un canal public, les ondes hertziennes, pour véhiculer des communications privées. La même remarque s'applique aux réseaux sans fil en général, par exemple les réseaux de type Wifi utilisés pour l'accès à Internet.

Garantir la fiabilité des nouvelles technologies au grand public nécessite donc l'usage de mécanismes de sécurité adéquats. La réponse aux questions évoquées ci-dessus est en particulier du ressort de la cryptographie.

1.1 Cryptographie et protocoles cryptographiques

1.1.1 Primitives et fonctionnalités

À l'origine, la cryptographie ou « art des codes secrets » avait pour objet exclusif les procédés de chiffrement. De nos jours, cette discipline scientifique s'intéresse à bien d'autres primitives cryptographiques, par exemple la signature électronique, les fonctions de hachage, la génération d'aléas... etc.

[1] Ceci est encore visible dans le nom utilisé « Inter-net ».

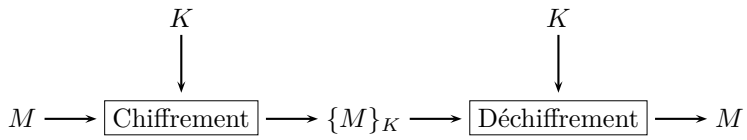

FIG. 1.1 – Chiffrement symétrique

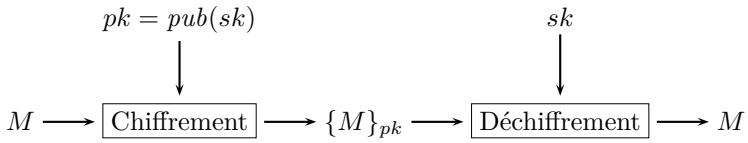

FIG. 1.2 – Chiffrement asymétrique

Ces primitives, ainsi que d'autres algorithmes spécifiques, permettent de réaliser diverses tâches cryptographiques de plus haut niveau, appelées *fonctionnalités* [MvV96]. Parmi les fonctionnalités plus courantes, on peut citer l'authentification d'un utilisateur sur un réseau, le stockage de données sécurisé, la distribution de clés, le vote électronique, ou encore le calcul distribué sur un réseau public.

L'un des principes fondamentaux de la cryptographie moderne, énoncé par Kerkhoff (1883), est de supposer que tous les algorithmes utilisés sont potentiellement connus de l'attaquant. Ainsi, sauf exception, la sécurité d'un système repose uniquement sur le secret de certains arguments donnés aux algorithmes, que l'on appelle traditionnellement des *clés*.

Parmi les primitives cryptographiques les plus utilisées, on compte de multiples variantes de chiffrement et de signature électronique. Par définition, un algorithme de *chiffrement* permet de coder des messages quelconques sous une forme inintelligible pour quiconque ne possède pas la clé de déchiffrement. On parle de *chiffrement symétrique* lorsque les clés de chiffrement et de déchiffrement sont identiques, et de *chiffrement asymétrique* dans le cas contraire (figures 1.1 et 1.2).

L'intérêt du chiffrement asymétrique réside dans la possibilité de diffuser la clé de chiffrement sans renoncer à la confidentialité des messages. Les clés de chiffrement et de déchiffrement sont alors appelées respectivement *clé publique* et *clé privée*. Pour simplifier, nous supposons sans perte de généralité que les clés publiques sont reliées à leurs clés privées respectives par une certaine fonction *pub* calculable efficacement (mais difficile à inverser !).

Une autre primitive classique est la *signature électronique* (figure 1.3). Cette primitive vise à garantir qu'un message signé n'a pas été modifié et qu'il émane bien de la personne signataire. Pour cela, chaque identité dispose donc d'une clé de signature, *privée*, et d'une clé de vérification, *publique*.

L'équivalent symétrique de la signature est constitué par les *codes d'authentification de message* (ou encore MACs pour *message authentication codes*) (figure 1.4). Les clés de vérification et de calcul d'un code d'authentification se confondent alors en une même clé privée.

D'une manière générale, une primitive est qualifiée de *probabiliste* lorsqu'elle utilise une source additionnelle d'aléa. (Ainsi, deux chiffrements probabilistes d'un

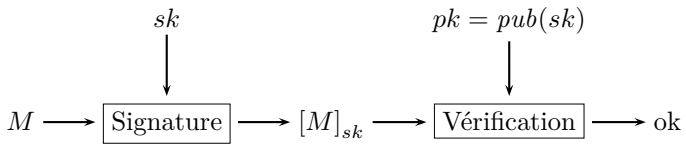

FIG. 1.3 – Signature

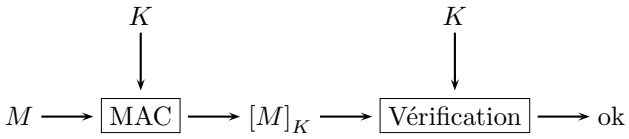

FIG. 1.4 – Code d'authentification de message (MAC)

même message donneront des résultats différents avec grande probabilité.) On parle de primitive *déterministe* dans le cas contraire.

Les fonctionnalités auxquelles nous nous intéressons dans cette thèse sont celles mettant en jeu un réseau de communication. Par exemple, l'*authentification* d'un utilisateur du réseau consiste à garantir (au minimum) son identité vis-à-vis d'un ou plusieurs agents avec lesquels il communique.

La difficulté de réaliser de telles fonctionnalités provient du fait que le réseau est susceptible d'être compromis, voire entièrement contrôlé par l'attaquant. Ainsi, en l'absence de réseau fiable, on doit considérer par précaution [NS78] que l'attaquant est en mesure

– d'intercepter et d'analyser tous les messages transitant sur le réseau,
– d'émettre les messages de son choix (à la place ou en plus des messages réels).

Cet état de fait justifie l'usage de la cryptographie dans les échanges de messages.

1.1.2 Protocoles cryptographiques

Un *protocole cryptographique* est un ensemble de programmes susceptibles de communiquer sur le réseau afin de réaliser une certaine fonctionnalité (cryptographique). Dans la typologie habituelle, chaque programme correspond à un *rôle* du protocole. Les *agents*, c'est-à-dire les machines ou les personnes susceptibles de mettre en œuvre le protocole peuvent jouer plusieurs rôles simultanément.

Lorsqu'un *agent honnête* endosse un rôle, une instance du programme correspondant au rôle, appelée également *processus*, s'exécute sur la machine de l'agent en utilisant ses différentes données personnelles, par exemple son identité et les clés secrètes qu'il détient.

À l'inverse, les *agents malhonnêtes* ne sont pas obligés de suivre un rôle du protocole. Dans la suite du document, nous nous plaçons dans le scénario où un ou plusieurs agents malhonnêtes sont utilisés par un attaquant global contrôlant le réseau de communication.

Dans la plupart des protocoles, les processus sont groupés naturellement en *sessions* : chaque session correspond à une instance de la fonctionnalité désirée du

protocole.

Exemple 1.1. À titre d'illustration, considérons une version simplifiée du protocole Denning-Sacco à clé publique [DS81, DJ04a]. Ce protocole entre deux rôles A et B permet à (l'agent jouant le rôle de) A de demander à B l'envoi d'un certain secret s_{AB} relatif à A et B. (Le protocole Denning-Sacco [DS81] original est destiné à la distribution de clés.)

Les échanges de messages entre A et B pour une session normale de ce protocole s'écrivent :

$$\begin{array}{lll} 0. & A \to B : & A,\ \{\,[K]_{sk_A}\,\}_{pk_B} \\ 1. & B \to A : & \{s_{AB}\}_K \end{array}$$

où

- le terme A désigne (un encodage de) l'identité de A,
- $[\cdot]_{sk_A}$ désigne l'opération de signature par la clé secrète de A,
- $\{\cdot\}_{pk_B}$ désigne le chiffrement par la clé publique de B, et
- $\{\cdot\}_K$ désigne le chiffrement par une clé symétrique K.

Plus précisément, la session se déroule de la manière suivante :

- A fabrique une clé fraîche K, calcule le premier message et l'envoie à B ;
- B tente de déchiffrer le message reçu avec sa clé privée sk_B ; en cas de succès, il vérifie que le clair obtenu est une signature valide de A en utilisant la clé publique de A ; si c'est le cas, il récupère K, construit le second message $\{s_{AB}\}_K$ et l'envoie à A.

Ce protocole peut sembler raisonnable au premier abord. Pourtant il possède une faille logique comparable à celle découverte par G. Lowe dans le protocole Needham-Shroeder [AN96, Low96, DJ04a].

En effet, l'attaquant I est susceptible de récupérer le secret s_{AB} à l'issue de la trace suivante :

$$\begin{array}{lll} 0. & A \to I : & A,\ \{\,[k_{AI}]_{sk_A}\,\}_{pk_I} \\ 0'. & I(A) \to B : & A,\ \{\,[k_{AI}]_{sk_A}\,\}_{pk_B} \\ 1. & B \to I(A) : & \{s_{AB}\}_{k_{AI}} \end{array}$$

Autrement dit, si un agent A entame une session avec I (c'est-à-dire n'importe quel agent malhonnête), alors I est capable ensuite d'exécuter une session complète avec B en se faisant passer pour A. La cause de cette attaque est ici le manque d'informations dans le message signé $[k_{AI}]_{sk_A}$: l'attaquant est en mesure de réutiliser ce message dans une autre session en tant qu'initiateur.

Propriétés des protocoles

Dans l'exemple précédent, l'attaque porte sur le secret de la donnée s_{AB}. D'une manière générale, la sécurité du protocole s'entend en fonction des différentes propriétés que le protocole est censé garantir.

Parmi les propriétés de sécurité les plus couramment étudiées, on compte

- le secret des données privées (dans un sens plus ou moins fort selon les cas),
- l'authentification d'un acteur ou d'un message du protocole,
- la résistance d'une donnée privée aux attaques par dictionnaire,
- la conformité du protocole à une certaine fonctionnalité idéale.

Plus rarement, des propriétés formulées en termes de jeux, comme l'équité, la non-répudiation, l'absence de coercition... etc sont également étudiées dans le cadre des protocoles de signature de contrat et de vote électronique.

Enjeux de l'analyse automatique de protocoles

L'exemple précédent illustre la difficulté de concevoir un protocole cryptographique. En particulier, des attaques comme celle décrite plus haut s'avèrent particulièrement difficiles à déceler sur des protocoles complexes du fait de l'interaction possible entre différentes sessions en parallèle.

En outre, les propriétés de sécurité en elles-mêmes étant rarement modulaires (*i.e.* la composition de plusieurs mécanismes sûrs n'est pas forcément sûre), prouver la correction d'un protocole directement à la main peut s'avérer extrêmement ardu.

Ainsi, de nombreux travaux visent depuis une vingtaine d'années à élaborer des outils d'analyse de protocoles cryptographiques (voir paragraphe suivant). Étant donnée la difficulté du problème, ces analyses sont généralement menées dans des modèles dits *symboliques* (ou encore *logiques*, *formels*), formulés de manière abstraite vis-à-vis de l'implémentation des primitives cryptographiques.

Le prix à payer pour cette abstraction est que certaines attaques sont susceptibles d'échapper à l'analyse : celles tirant justement parti des algorithmes choisis pour chaque primitive. Pourtant, de telles attaques sont prises en compte dans le formalisme standard utilisé pour les preuves de sécurité (manuelles) en cryptographie.

Partant de ce constat, un effort important de recherche a débuté il y a quelques années [LMMS98, AR02] dans le but de développer des outils de preuve valables dans le formalisme cryptographique standard, et si possible toujours automatiques. Une manière d'atteindre ce but est par exemple de justifier les outils symboliques existants, c'est-à-dire de montrer que l'absence d'attaques logiques sur un protocole implique l'existence d'une preuve de sécurité dans le modèle cryptographique.

1.2 Analyse symbolique de protocoles

Les premiers modèles symboliques sont généralement attribués à R. Needham et M. Schroeder [NS78], et à D. Dolev et A. Yao [DY83]. Ces modèles consistent en une formalisation abstraite des protocoles dans le but d'identifier certaines failles de conception, dites *logiques*. L'exemple 1.1 constitue l'illustration d'une telle faille logique, c'est-à-dire d'une attaque dont la réalisation ne dépend pas du choix de l'implémentation des primitives.[2]

Plus précisément, les modèles symboliques se caractérisent par les faits suivants :

- les messages du protocole sont représentés de manière syntaxique par des *termes* (en programmation, on dirait des expressions non évaluées) ;
- l'attaquant est autorisé à construire et à déconstruire les messages au moyen d'un ensemble fini d'opérations algébriques, par exemple : former la paire de deux messages, ou déchiffrer un message s'il détient la bonne clé.

L'intérêt d'un tel cadre abstrait est de se prêter volontiers à l'analyse automatique des propriétés de sécurité, notamment en ce qui concerne le secret des données et l'authentification. À cet égard on peut distinguer deux familles d'approches.

[2] D'autres exemples de failles logiques, portant notamment sur des propriétés de sécurité plus complexes, sont abordés dans le chapitre 3.

Recherche d'attaques logiques

Une première famille regroupe les méthodes de recherche d'attaques en un nombre borné de sessions. Les premiers outils, fondés sur la vérification de modèle (*model-checking*) [Low96, MMS97], consistaient à explorer l'ensemble de l'espace des attaques possibles, pour un nombre borné de sessions et pour des messages de taille bornée.

A. Huima [Hui99a] suggéra le premier que borner la taille des messages n'était pas nécessaire pour peu que l'on utilise un algorithme de résolution de contraintes symboliques ad hoc. Depuis cette voie a été largement explorée [AL00, MS01, Bor01], notamment avec la prise en compte récente de primitives ayant des propriétés algébriques comme le OU exclusif ou l'exponentiation modulaire à la Diffie-Hellman [BB05, DJ04a, CKRT03a, MS03, CKRT03b, CLS03, DLLT06, GRV05, CR05]. D'autres travaux ont été également entrepris afin de prendre en compte des propriétés de sécurité fines comme la résistance aux attaques par dictionnaire [CMAFE03, DJ04b], les équivalences de processus (finis) [DSV03], et les propriétés de jeux liées à la signature de contrat électronique [KR02, KK05].

Preuve symbolique

Une seconde famille d'approches regroupe les méthodes de *preuve symbolique*. Ces méthodes s'appliquent à un nombre non borné de sessions au prix d'un algorithme approché, c'est-à-dire susceptible de refuser un protocole sans faille logique. Cette approximation est rendue nécessaire par l'indécidabilité du problème en général [EG83].

Tandis que les premières méthodes reposaient sur un assistant de preuve [Pau98, Bol97], d'autres techniques manuelles ont été proposées depuis, notamment à base de systèmes de type dans une algèbre de processus [Aba99], d'une logique spécialisée [BAN90, DMP03] ou d'autres formalismes [THG99].

Enfin, de nombreux outils de preuve automatiques ont été élaborés, reposant sur des techniques diverses, notamment :

- la recherche d'invariants inductifs [Mea99, CMR01],
- de l'analyse du flot de contrôle (*control flow analysis*) [BDNN02, ZD06],
- les automates d'arbre [Mon03, Gou00, CCM01, ZD06], et
- les clauses de Horn [Bla01, Gou04, AB05a, GRV05] avec plus récemment une extension aux propriétés d'équivalence observationnelle dans un langage de processus cryptographique [BAF05].

Le cadre symbolique dispose donc d'un nombre important d'outils d'analyse automatiques. Bien que le problème soit indécidable dans le cas général (et souvent (co-)NP-complet pour un nombre borné de sessions), les analyseurs les plus efficaces montrent de bonnes performances pratiques : par exemple Proverif [Bla01, BAF05] répond le plus souvent en quelques secondes.

Les principales limites de l'approche symbolique résident sans surprise dans l'abstraction réalisée au moment de la modélisation :

- d'une part, il reste difficile de modéliser les protocoles qui ne sont pas conçus de manière modulaire, c'est-à-dire pour lesquels on en peut pas voir les primitives comme des « boîtes noires » ;

– d'autre part, il s'avère qu'un protocole sans faille logique n'est pas nécessairement sûr d'un point de vue cryptographique, à plus fortes raisons si des primitives cryptographiques de sécurité insuffisante sont employées[3].

Ce dernier constat est à l'origine d'un nombre important de travaux sur la justification calculatoire de modèles symboliques.

1.3 Lien entre modèles logiques et cryptographie

À la différence des modèles symboliques, les modèles cryptographiques (ou calculatoires, pour « *computational* ») tiennent compte de la description mathématique des différentes primitives. Ainsi, les messages sont modélisés comme des chaînes de bits, c'est-à-dire des mots sur l'alphabet $\{0, 1\}$.

De plus, l'attaquant cryptographique n'est pas limité à certaines opérations a priori, mais est libre d'effectuer n'importe quel calcul faisable en temps polynomial randomisé. Dans ce formalisme, les preuves de sécurité sont établies par réduction au sens de la théorie de la complexité. Autrement dit, pour montrer la sécurité d'un protocole P, on montre que s'il existe une attaque sur P, alors il existe également une attaque sur un problème de référence R. On en déduit alors que P est sûr sous l'hypothèse (largement étudiée et admise) que R est sûr.

Dans le cas d'un protocole, élaborer une telle réduction est une tâche complexe et source d'erreurs. La question de faire le lien entre les deux familles de modèles, symboliques et cryptographiques, que nous avons évoquées présente donc un intérêt manifeste : idéalement on souhaiterait conserver les avantages des deux approches pour obtenir des outils d'analyse automatiques de protocoles, valables dans le formalisme cryptographique standard.

À cet égard, les travaux précurseurs de P. Lincoln *et al.* [LMMS98] et de M. Abadi et de P. Rogaway [AR00, AR02] ont suscité de nombreuses voies de recherche.

Indistinguabilité entre données (cas passif)

Le résultat de M. Abadi et de P. Rogaway [AR00, AR02] constitue le premier exemple de critère logique possédant une justification cryptographique. Considérant des termes construits sur une signature comprenant la paire, le chiffrement symétrique, des données privées (les clés) et des constantes, les auteurs montrent que si deux termes vérifient un certain critère formel alors les données concrètes ainsi représentées sont indistinguables par tout attaquant polynomial.

Ce résultat a depuis été étendu de différentes manières : par une étude de la complétude [MW04a], par du chiffrement déterministe [Lau02], par différentes variantes de chiffrement (révélant la taille des messages ou les clés utilisées, ou encore à base de One-Time Pad) [ABS05], par les clés composées [LC03] et enfin par les cycles de chiffrement [AHBS05].

Justification de modèles formels dans le cas actif

Les premiers résultats justifiant un modèle formel dans le cas actif sont dus à M. Backes, B. Pfitzmann et M. Waidner [BPW03a, BPW03b, BP04]. Ces travaux

[3] B. Warinschi [War03] considère le contre-exemple du protocole Needham-Schroeder-Lowe à clés publiques lorsque le chiffrement est implémenté par la primitive Elgamal.

montrent la sûreté cryptographique d'une librairie de primitives abstraites incluant les chiffrements symétrique et asymétrique et la signature asymétrique. L'expression *sûreté cryptographique* est prise ici dans un sens fort lié à la notion de simulabilité (*simulatability*) ; en particulier toute attaque sur le système concret doit correspondre à une attaque « semblable » sur le système abstrait. Des résultats similaires, pour une notion de sûreté moins forte basée sur les traces d'exécution mais des modèles logiques plus facilement automatisables, ont été obtenus par D. Micciancio et B. Warinschi [MW04b], puis simultanément par V. Cortier et B. Warinschi [CW05] et par R. Janvier, Y. Lakhnech et L. Mazaré [JLM05]. Récemment, des outils de preuve automatiques ont été adaptés pour s'appliquer également aux notions plus fortes de composabilité universelle (*Universal Composability*) [CH06] et de simulabilité [BL06].

Formalismes logiques spécifiques

Dans une perspective différente, d'autres travaux tentent de formaliser directement les raisonnements cryptographiques dans un cadre logique original. Ainsi, l'algèbre de processus probabilistes de P. Lincoln *et al.* [LMMS98] a donné lieu à plusieurs raffinements successifs par P. Mateus *et al.* [MMS03], J. Mitchell *et al.* [MRST01] et A. Ramanathan *et al.* [RMST04]. Dans le même but, A. Datta *et al.* [DDM+05] proposent une logique dédiée à la preuve de protocoles et justifiée au niveau cryptographique. Toutefois, ces différents formalismes logiques ne disposent pas d'outil automatique à l'heure actuelle.

Automatisation des preuves cryptographiques

Récemment, une voie de recherche prometteuse consistant à automatiser directement la recherche de preuves cryptographiques a été initiée par P. Laud [Lau04], dans le cas du chiffrement symétrique pour un nombre fini de sessions, et par B. Blanchet [Bla06, BP06], dans un cadre général. Plus précisément, B. Blanchet [Bla06] a montré à travers l'outil Cryptoverif [Blaa] qu'il est possible d'automatiser en grande partie la recherche de preuves dans le formalisme des jeux cryptographiques à la V. Shoup [Sho04].

1.4 Contributions et contenu de la thèse

Les contributions de cette thèse s'articulent autour des deux thématiques de recherche évoquées plus haut concernant les protocoles cryptographiques.

Attaques par dictionnaire et propriétés d'équivalence

En 2003, lorsque cette thèse a débuté, plusieurs modèles symboliques distincts coexistaient pour les attaques par dictionnaire [Low04, Coh02, CMAFE03, DJ04b]. Bien que reposant sur des intuitions similaires décrites par G. Lowe [Low04], ces définitions s'avéraient mutuellement incomparables, sans qu'il semblât alors possible de les départager et encore moins de les généraliser à des primitives cryptographiques avec propriétés algébriques, comme par exemple le OU exclusif.

Au cours de l'année suivante, le cadre général du pi-calcul appliqué [AF01] suscita une nouvelle modélisation symbolique des attaques par dictionnaire repo-

sant sur la notion d'équivalence statique [AF01] et due indépendamment à Fournet [Fou02] et à R. Corin, J. Doumen et S. Etalle [CDE04]. L'intérêt de cette définition était de proposer pour la première fois un critère général, paramétré par le choix d'une signature et d'une théorie équationnelle. Toutefois, l'article original de R. Corin *et al.* [CDE04] ne mentionne que le cas d'un intrus passif. De plus, cette définition ne bénéficiait initialement d'aucun outil d'analyse.

Plus récemment, B. Blanchet, M. Abadi et C. Fournet [BAF05] ont formulé un critère approché pour la sécurité contre les attaques par dictionnaire dans le « nouveau » formalisme équationnel. Ce critère est valable dans le cas actif, pour un nombre non borné de sessions. L'implémentation, dans l'outil Proverif [Blab], montre de bonnes performances pratiques malgré la possibilité de non-terminaison.

À cette époque, l'existence d'un algorithme de décision (*i.e.* exact) pour les attaques par dictionnaire en un nombre borné de sessions selon la nouvelle définition générale (par exemple pour les théories équationnelles usuelles du chiffrement) restait donc un problème ouvert — le cas passif étant quant à lui subsumé par les travaux d'Abadi et Cortier [AC04].

Dans cette optique, une première contribution de cette thèse est de clarifier la notion d'attaque par dictionnaire dans le cas d'un intrus actif, en s'inspirant de la définition générale de R. Corin *et al.* [CDE04] et du critère proposé par B. Blanchet *et al.* [BAF05]. Dans le chapitre 3, nous proposons en effet un langage proche du picalcul appliqué étendu par des phases [BAF05], permettant (entre autres) d'énoncer la résistance aux attaques par dictionnaire de manière naturelle. En particulier, nous montrons que, dans ce cadre, la notion d'attaque par dictionnaire ne dépend pas du choix de l'équivalence observationnelle pour le langage (parmi les notions usuelles).

Dans le même chapitre, nous donnons ensuite une sémantique symbolique à ce langage de processus, et montrons que le problème de la résistance aux attaques par dictionnaire pour un nombre borné de sessions se ramène à celui de la résolution de systèmes de contraintes symboliques bien choisis — en l'occurrence à base d'unification équationnelle du second ordre.

Nous nous efforçons de résoudre ces systèmes de contraintes dans le chapitre 4 pour la famille des théories sous-terme-convergentes (définie au chapitre 2). Une première version de l'algorithme de résolution détaillé ici a été présentée à la conférence CCS'05 [Bau05].

Ensemble, les résultats des chapitres 3 et 4 décrivent ainsi la première procédure exacte, dans un cadre général, pour décider la sécurité contre les attaques par dictionnaire en un nombre borné de sessions. Plus exactement, nous montrons que ce problème est dans co-NP pour toute théorie sous-terme-convergente (finiment présentée).

Soulignons que cette procédure s'applique à davantage de propriétés de sécurité. Outre les propriétés habituelles de secret simple et d'authentification, nous traitons en effet le cas d'une équivalence observationnelle à base de bi-processus inspirée de [BAF05] et généralisant la définition des attaques par dictionnaire. Cette équivalence implique en particulier les équivalences observationnelles définissables de la manière usuelle dans une algèbre de processus.

Justification cryptographique de modèles symboliques équationnels

Comme nous l'avons évoqué, justifier un modèle symbolique préexistant dans le formalisme cryptographique standard est une manière naturelle d'obtenir un outil automatique de preuve cryptographique.

En elle-même, cette question présente en outre un certain intérêt théorique : dans l'hypothèse (réaliste) où les attaques logiques sur un protocole sont toujours réalisables au niveau calculatoire, un modèle symbolique logiquement complet (*i.e.* sans approximations) est sûr dans un modèle cryptographique si et seulement si tout protocole sans faille logique est prouvablement sûr dans le même modèle. Ainsi, lorsqu'elle est vraie, cette propriété témoigne d'une forme de complétude du modèle cryptographique sous-jacent.

Nous avons vu plus haut que la plupart des modèles logiques récents sont (au moins partiellement) équationnels [AF01, BB05, DJ04a, CKRT03a, MS03, CKRT03b, CLS03, DLLT06, CR05, GRV05, BAF05, ZD06]. En effet, une des motivations principales du formalisme équationnel était initialement de relâcher l'hypothèse du *chiffrement parfait*, autrement dit de se rapprocher du modèle calculatoire en tenant compte de certaines propriétés algébriques des primitives : par exemple, la commutation éventuelle des opérations de chiffrement et d'appariement (propriété d'« homomorphisme »).

De plus, le formalisme équationnel, tel qu'il est utilisé notamment dans le pi-calcul appliqué [AF01], s'avère bien adapté pour exprimer les propriétés logiques d'équivalence, dont on peut voir les attaques par dictionnaire comme un cas particulier [Fou02, CDE04].

Or, à l'exception des travaux récents de B. Blanchet [Bla06, BP06], poursuivant une approche différente, les travaux mentionnés ci-dessus pour la justification de modèles symboliques concernent tous des modèles logiques à base de termes libres, c'est-à-dire sans équations. Lorsque cette thèse a débuté, se posait donc la question naturelle de justifier un modèle symbolique avec équations dans le formalisme cryptographique.

À cet égard, l'une des contributions de cette thèse (chapitre 5) est de proposer un cadre général pour comparer modèles symboliques (équationnels) et modèles calculatoires vis-à-vis de l'indistinguabilité des données — au sens du paragraphe précédent et de M. Abadi et de P. Rogaway [AR00]. Par « général », on entend ici valable pour un jeu de primitives et une théorie équationnelle arbitraires. Notamment, nous présentons et discutons un critère suffisant pour la sûreté calculatoire d'un modèle symbolique arbitraire. Nous étudions également l'exemple des groupes abéliens et montrons que la sûreté calculatoire de ce modèle symbolique implique l'hypothèse de Diffie-Hellman décisionnelle, ainsi que l'hypothèse RSA (moyennant l'extension de la signature par un symbole libre).

Dans le chapitre 6, nous développons deux applications du chapitre 5. Une première application, élémentaire, concerne la théorie du OU exclusif. Le reste du chapitre est consacré à la preuve de sûreté d'une théorie plus complexe modélisant différentes formes de chiffrement : symétrique, asymétrique et symétrique déterministe surjectif (dit *cipher* ou *permutation pseudo-aléatoire*).

Ce dernier résultat permet en particulier de justifier dans le cas passif la définition générale des attaques par dictionnaire utilisée dans le chapitre 3 et due à C. Fournet [Fou02] et à R. Corin *et al.* [CDE04]. Ainsi, cette définition des attaques

par dictionnaire est la première à bénéficier à la fois

- d'un résultat de décidabilité pour un nombre borné de sessions (chapitres 3 et 4 de cette thèse),
- d'une procédure approchée pour un nombre non-borné de sessions [BAF05], et
- d'une justification calculatoire dans le cas passif (chapitres 6) pour un jeu de primitives donné.

Les résultats présentés dans les chapitres 5 et 6 ont été obtenus en collaboration avec Véronique Cortier et Steve Kremer d'une part (chapitre 5 et section 6.1), et avec Martín Abadi et Bogdan Warinschi d'autre part (sections 6.2 et suivantes). Ils ont fait l'objet de publications, respectivement aux conférences ICALP'05 [BCK05] et FOSSACS'06 [ABW06].

Chapitre 2

Préliminaires : logique équationnelle et réécriture

Sommaire

Ce chapitre est consacré en premier lieu (sections 2.1 à 2.4) à un rappel de logique équationnelle et de théorie de la réécriture en vue des parties suivantes. La plupart des définitions et des notations introduites ici sont standards, à l'exception (semble-t-il) des notions de domaine et de substitution partielle définies dans la section 2.2.

Dans la section 2.5, nous décrivons une caractérisation des symboles libres d'une théorie à l'aide d'une notion adéquate de remplacement modulo E. Nous rappelons également la notion de système de réécriture canonique dans la section 2.6. Enfin, nous terminons ce chapitre (section 2.7) par la définition d'une famille de théories équationnelles jouant un rôle notable pour les applications cryptographiques des chapitres 4 et 6, appelées théories sous-terme-convergentes.

2.1 Termes du premier ordre

Une *signature du premier ordre* (ou simplement *signature* par la suite) est la donnée d'un ensemble de *sortes* (ou *types*) $\mathcal{T}$, d'un ensemble de *symboles de fonction* $\mathcal{F}$ et d'une fonction, notée ar, qui à tout symbole $f \in \mathcal{F}$ associe son *arité* $\mathrm{ar}(f) = \tau_1 \times \cdots \times \tau_k \to \tau$, où $\tau_1, \ldots, \tau_k, \tau \in \mathcal{T}$ sont des sortes et $k \geq 0$ est le *nombre d'arguments* de f. Une *constante de sorte* τ est un symbole de fonction d'arité $(\to \tau)$.

Soit $\mathcal{X}$ un ensemble de *variables*. On suppose que chaque variable est munie d'une sorte et qu'il existe un nombre infini de variables de chaque sorte. L'ensemble des *termes* construits sur la signature $(\mathcal{T}, \mathcal{F})$ et sur les variables de $\mathcal{X}$ est défini inductivement par

$$\begin{array}{rcll} T & ::= & & \text{terme de sorte } \tau \\ & | & x & \text{variable } x \in \mathcal{X} \text{ de sorte } \tau \\ & | & f(T_1, \ldots, T_k) & \text{application d'un symbole } f \in \mathcal{F} \end{array}$$

où, dans le dernier cas, étant donné $\mathrm{ar}(f) = \tau_1 \times \cdots \times \tau_k \to \tau$, chaque T_i est un terme de sorte τ_i. Chaque terme T possède donc une unique sorte notée $\mathrm{sort}(T)$.

Lorsque $\mathcal{T}$ et ar sont clairs, on note $\mathcal{F}[\mathcal{X}]$ l'ensemble des termes ainsi définis, muni de la structure de $\mathcal{F}$-algèbre (multi-sortée) évidente.

Plus généralement, étant donné un sous-ensemble de symboles $\mathcal{F}' \subseteq \mathcal{F}$, et un ensemble de termes $A \subseteq \mathcal{F}[\mathcal{X}]$, on note $\mathcal{F}'[A]$ la sous-$\mathcal{F}'$-algèbre engendrée par A dans $\mathcal{F}[\mathcal{X}]$, c'est-à-dire le plus petit ensemble contenant A et stable par application (bien sortée) de symboles $f \in \mathcal{F}'$.

On note $\mathrm{var}(T)$ l'ensemble des variables contenues dans un terme T. Un terme T est dit *clos* si $\mathrm{var}(T) = \emptyset$. La notation var est étendue aux ensembles et n-uplets de termes de la manière évidente.

On note $\mathrm{pos}(T)$ l'ensemble des *positions* d'un terme T, vu comme un arbre étiqueté. Formellement, $\mathrm{pos}(T)$ est un ensemble de mots sur l'alphabet $\{1, 2 \ldots\}$, défini inductivement par

$$\mathrm{pos}(x) \triangleq \{\epsilon\} \qquad \mathrm{pos}(f(T_1, \ldots, T_n)) \triangleq \{\epsilon\} \cup \bigcup_{i=1}^{n} i \cdot \mathrm{pos}(T_i),$$

où ϵ désigne le mot vide, et $\cdot$ la concaténation de mots. Soit p une position de T. On note $T|_p$ le sous-terme de T en position p, et $T[T']_p$ le terme obtenu en remplaçant la p-ième position de T par un terme T' de sorte $\mathrm{sort}(T') = \mathrm{sort}(T|_p)$. L'ensemble des sous-termes de T est noté $\mathrm{st}(T)$:

$$\mathrm{st}(T) \triangleq \{T|_p \mid p \in \mathrm{pos}(T)\}.$$

Par extension, étant donné une équation (orientée) $T \doteq S$, on définit les notations $(T \doteq S)|_p$ et $(T \doteq S)[T']_p$ de manière similaire.

2.2 Substitutions

Une *substitution* est une fonction σ de $\mathcal{X}$ vers $\mathcal{F}[\mathcal{X}]$ bien sortée, c'est-à-dire telle que $\forall x \in \mathcal{X}$, $\mathrm{sort}(\sigma(x)) = \mathrm{sort}(x)$. Plus généralement, on appelle *substitution partielle* toute fonction bien sortée θ d'un ensemble de variables $\mathrm{dom}(\theta) \subseteq \mathcal{X}$, appelé *ensemble de départ* (ou *domaine*) de θ, vers $\mathcal{F}[\mathcal{X}]$.

On appelle *support de* θ l'ensemble $\mathrm{supp}(\theta) \triangleq \{x \in \mathrm{dom}(\theta) \mid \theta(x) \neq x\}$.[1] Par la suite, on considère uniquement des substitutions partielles (ou totales) à support

[1] Cette dénomination (utilisée également par Y. Chevalier et M. Rusinowitch [CR05]) n'est pas standard en réécriture, mais l'est en algèbre, par exemple dans l'étude des permutations, où le mot domaine a déjà le sens que nous lui donnons. Plus bas, nous suivons également la notion habituelle d'image d'une fonction.

fini. Si $\mathrm{supp}(\theta) = \{x_1, \ldots, x_n\}$, $\mathrm{dom}(\theta) = \mathcal{Y}$ et $\forall i$, $\theta(x_i) = T_i$, on note

$$\theta = \{x_1 \mapsto T_1, \ldots, x_n \mapsto T_n\}_{\mathcal{Y}}$$

ou plus simplement $\theta = \{x_1 \mapsto T_1, \ldots, x_n \mapsto T_n\}$ lorsque l'ensemble de départ de θ est clair. L'*ensemble des valeurs* (ou *image*) de θ est défini par $\mathrm{im}(\theta) \triangleq \theta(\mathrm{dom}(\theta))$. Une substitution partielle θ est *close* si $\mathrm{var}(\mathrm{im}(\theta)) = \emptyset$.

L'*union* de deux substitutions partielles θ et θ' de domaines disjoints est notée $\theta \uplus \theta' : \mathrm{dom}(\theta \uplus \theta') \triangleq \mathrm{dom}(\theta) \uplus \mathrm{dom}(\theta')$, $(\theta \uplus \theta')(x) \triangleq \theta(x)$ si $x \in \mathrm{dom}(\theta), \theta'(x)$ sinon. La *restriction* d'une substitution partielle θ à un ensemble de variables $\mathcal{Y} \subseteq \mathrm{dom}(\theta)$ est notée $\theta|_{\mathcal{Y}} \triangleq \{x \mapsto \theta(x)\}_{x \in \mathcal{Y}}$. Inversement, lorsque $\mathrm{dom}(\theta) \subseteq \mathcal{Y}$, la notation $\theta\|_{\mathcal{Y}}$ désigne le prolongement de θ à $\mathcal{Y}$, défini par $\theta\|_{\mathcal{Y}} \triangleq \theta \uplus \{x \mapsto x\}_{x \in \mathcal{Y} - \mathrm{dom}(\theta)}$.

L'*application* d'une substitution partielle θ à un terme T tel que $\mathrm{var}(T) \subseteq \mathrm{dom}(\theta)$ est notée $T\theta$:

$$x\theta \triangleq \theta(x) \qquad f(T_1, \ldots, T_n)\,\theta \triangleq f(T_1\theta, \ldots, T_n\theta).$$

Cette notation est généralisée aux ensembles et n-uplets de termes de la manière naturelle.

Étant données deux substitutions partielles θ et λ telles que $\mathrm{var}(\mathrm{im}(\theta)) \subseteq \mathrm{dom}(\lambda)$, on note $\theta\lambda = \lambda \circ \theta$ la *composition* de θ et λ : $\mathrm{dom}(\theta\lambda) \triangleq \mathrm{dom}(\theta)$, et $\forall x \in \mathrm{dom}(\theta)$, $x(\theta\lambda) \triangleq (x\theta)\lambda$. Ainsi, pour tout T tel que $\mathrm{var}(T) \subseteq \mathrm{dom}(\theta)$, on a $(T\theta)\lambda = T(\theta\lambda)$.

Une substitution σ est *idempotente* ssi $\sigma\sigma = \sigma$, c'est-à-dire $\mathrm{supp}(\sigma) \cap \mathrm{im}(\sigma) = \emptyset$. (Notons que d'une manière générale, $\mathrm{supp}(\sigma) \cup \mathrm{im}(\sigma) \supseteq \mathcal{X}$.)

Un *renommage* est une substitution partielle ρ telle que $\mathrm{im}(\rho) \subseteq \mathcal{X}$ et ρ induise une bijection entre $\mathrm{dom}(\rho)$ et $\mathrm{im}(\rho)$. Tout renommage ρ admet donc un renommage inverse, noté ρ^{-1}, vérifiant $\mathrm{dom}(\rho^{-1}) = \mathrm{im}(\rho)$, $\mathrm{im}(\rho^{-1}) = \mathrm{dom}(\rho)$ et $\mathrm{supp}(\rho^{-1}) = \rho(\mathrm{supp}(\rho))$.

Un *unificateur* de deux termes T et S est une substitution σ telle que $T\sigma = S\sigma$. Un tel σ est dit *principal* (ou *plus général*) lorsque pour tout unificateur μ, $\sigma\mu = \mu$. On montre que si T et S sont unifiables, alors il existe un unificateur principal. De plus, si σ et μ sont deux unificateurs principaux de T et S, alors il existe un renommage ρ allant de $\mathrm{var}(\mathrm{im}(\sigma))$ vers $\mathrm{var}(\mathrm{im}(\mu))$ tel que $\sigma\rho = \mu$. On note $\mathrm{mgu}(T, S)$ un unificateur principal de T et de S (fixé) lorsqu'il existe.

2.3 Théories équationnelles et systèmes de réécriture

Une théorie équationnelle E sur une algèbre de termes $\mathcal{F}[\mathcal{X}]$ est un ensemble d'équations $T \doteq S$ (où $T, S \in \mathcal{F}[\mathcal{X}]$ sont de même sorte) clos par les règles de déduction de la logique équationnelle (table 2.1) au sens où $E \vdash T \doteq S$ implique $(T \doteq S) \in E$. Par construction, la relation binaire $=_E$ sur $\mathcal{F}[\mathcal{X}]$, définie par $T =_E S$ ssi $(T \doteq S) \in E$, est donc une congruence (relation réflexive, symétrique, transitive, stable par application de contexte), qui plus est, stable par substitution. Lorsque E est simplement un ensemble d'équations, $=_E$ désigne la relation équationnelle engendrée par E : $T =_E S$ ssi $E \vdash T \doteq S$.

$$\frac{T \doteq S \in E}{E \vdash T \doteq S} \text{ (axiome)}$$

$$\frac{}{E \vdash T \doteq T} \text{ (réflexivité)} \qquad \frac{E \vdash S \doteq T}{E \vdash T \doteq S} \text{ (symétrie)}$$

$$\frac{E \vdash S \doteq T \quad E \vdash T \doteq U}{E \vdash S \doteq U} \text{ (transitivité)}$$

$$\frac{\forall 1 \leq i \leq n,\ E \vdash T_i \doteq S_i \quad f(T_1 \dots T_n), f(S_1, \dots, S_n) \in \mathcal{F}[\mathcal{X}]}{E \vdash f(T_1, \dots, T_n) \doteq f(S_1, \dots, S_n)} \text{ (contexte)}$$

$$\frac{E \vdash S \doteq T}{E \vdash S\sigma \doteq T\sigma} \text{ (substitution)}$$

Tab. 2.1 – Déduction en logique équationnelle

Une sorte τ est dite *dégénérée pour* E si pour tous termes T et S de sorte τ, $T =_E S$, ou de manière équivalente, s'il existe deux variables distinctes x et y de sorte τ telles que $x =_E y$.

Une *règle de réécriture* est une expression $l \to r$ où l et r sont deux termes de même sorte. Un *système de réécriture* est un ensemble $\mathcal{R}$ de règles de réécriture (éventuellement infini). Étant donné un système de réécriture $\mathcal{R}$, on écrit $T \to_{\mathcal{R}} T'$ s'il existe une règle $l \to r \in \mathcal{R}$, une position p et une substitution σ telle que $T|_p = l\sigma$ et $T' = T[r\sigma]_p$. On note $\to^*_{\mathcal{R}}$ la clôture réflexive et transitive de $\to_{\mathcal{R}}$, et $=_{\mathcal{R}}$ sa clôture réflexive, symétrique et transitive. Nous utilisons également les notations $\to^+_{\mathcal{R}}$ et $\to^{\leq 1}_{\mathcal{R}}$ pour désigner respectivement la clôture transitive de $\to_{\mathcal{R}}$ et sa clôture réflexive.

Par le théorème de Birkhoff (1933), on sait que $=_{\mathcal{R}}$ est une congruence stable par substitution dont la théorie équationnelle correspondante est engendrée par $E(\mathcal{R}) = \{l \doteq r \mid (l \to r) \in \mathcal{R}\}$. Autrement dit, on a l'équivalence :

$$T =_{\mathcal{R}} S \Leftrightarrow E(\mathcal{R}) \vdash T \doteq S$$

Inversement, on peut orienter tout ensemble d'équations E en un système de réécriture $\mathcal{R}(E) = \{T \to S \mid (T \doteq S) \in E\}$ de sorte que

$$T =_{\mathcal{R}(E)} S \Leftrightarrow E \vdash T \doteq S.$$

Étant donné une théorie équationnelle E et un système de réécriture $\mathcal{R}$, $\to_{\mathcal{R}/E}$ désigne la relation $=_E \to_{\mathcal{R}} =_E$. On définit les relations $\to^*_{\mathcal{R}/E}$ et $=_{\mathcal{R}/E}$ comme ci-dessus. Un système $\mathcal{R}$ est *E-terminant* (ou *E-nœthérien*) ssi $\to_{\mathcal{R}/E}$ n'a pas de suite infinie de réductions $T_0 \to_{\mathcal{R}/E} T_1 \to_{\mathcal{R}/E} \dots T_n \to_{\mathcal{R}/E} \dots$ Il est *E-confluent* ssi pour tous $T \to^*_{\mathcal{R}/E} T_1$ et $T \to^*_{\mathcal{R}/E} T_2$, il existe T'_1 et T'_2 tels que $T_1 \to^*_{\mathcal{R}/E} T'_1$, $T_2 \to^*_{\mathcal{R}/E} T'_2$, et $T'_1 =_E T'_2$. Enfin, $\mathcal{R}$ est *E-convergent* ss'il est E-terminant et E-confluent.

Un terme T est une *forme $\mathcal{R}/E$-normale* lorsqu'il n'existe pas de terme T' tel que $T \to_{\mathcal{R}/E} T'$. (On dit aussi que T est *$\mathcal{R}/E$-normal* ou *$\mathcal{R}/E$-réduit.*) Les systèmes

de réécriture E-convergents sont caractérisés par le fait que tout terme T se réduit en une forme normale unique modulo E. La notation $T\downarrow_{\mathcal{R}/E}$ désigne alors une forme $\mathcal{R}/E$-normale de T (fixée arbitrairement).

On sait également par le Lemme de Newman qu'un système $\mathcal{R}$ est E-convergent ss'il est E-terminant et *localement E-confluent* au sens où pour tous $T \to_{\mathcal{R}/E} T_1$ et $T \to_{\mathcal{R}/E} T_2$ (en une seule étape), il existe T_1' et T_2' tels que $T_1 \to^*_{\mathcal{R}/E} T_1'$, $T_2 \to^*_{\mathcal{R}/E} T_2'$, et $T_1' =_E T_2'$.

Lorsque $=_E$ est la relation d'égalité, nous retrouvons les notions usuelles de terminaison, de confluence (locale), de convergence et de forme normale d'un système de réécriture.

La proposition suivante caractérise simplement les sortes dégénérées d'un système de règles E-convergent.

Proposition 2.1. *Soit E une théorie équationnelle et $\mathcal{R}$ un système de réécriture E-convergent. Notons $\mathcal{R} \cup E$ la théorie équationnelle correspondant à $=_{\mathcal{R}/E}$.*

Une sorte τ est dégénérée dans $\mathcal{R} \cup E$ ssi elle est dégénérée dans E.

Démonstration. Soit x, y deux variables distinctes de sorte τ. Comme $\mathcal{R}$ est E-terminant, x et y sont en forme E-normale. Si $x =_{\mathcal{R}/E} y$, par E-confluence, on a donc $x =_E y$ et τ est dégénérée dans E. La réciproque est évidente du fait que $(=_E) \subseteq (=_{\mathcal{R}/E})$. □

En particulier, un système de réécriture convergent $\mathcal{R}$ n'a donc pas de sorte dégénérée. Notons toutefois qu'une sorte τ peut être encore dégénérée en ce qui concerne les termes clos : pour tous $S, T \in \mathcal{F}[\emptyset]$ de sorte τ, $S =_{\mathcal{R}} T$. (Considérer une signature contenant une seule constante.)

2.4 Problèmes équationnels et complexité

Soit E une théorie équationnelle sur $\mathcal{F}[\mathcal{X}]$. Le *problème du mot* sur E est le problème algorithmique consistant, étant donnés deux termes clos $T, S \in \mathcal{F}[\emptyset]$, à décider si $T =_E S$.

Le *problème de l'unification modulo E* (ou encore *E-unification*) est le problème algorithmique consistant, étant donnés deux termes $T, S \in \mathcal{F}[\mathcal{X}]$, à décider s'il existe une substitution σ telle que $T\sigma =_E S\sigma$.

Pour étudier la complexité de ces deux problèmes — et plus généralement de tout problème algorithmique sur des termes —, il est nécessaire de spécifier une représentation concrète des n-uplets (ou ensembles) de termes sous forme de chaînes de bits. Cette représentation conditionne notamment la notion de taille pour une instance du problème.

Une première représentation possible est celle des n-uplets d'arbres étiquetés sans partage (voir figure 2.1). Lorsque la signature $(\mathcal{T}, \mathcal{F})$ et l'ensemble de variables $\mathcal{X}$ sont finis, une notion de taille de problème adaptée est alors la somme du nombre de noeuds et de feuilles utilisés pour représenter chaque terme. Étant donné un terme T, on note $|T| \triangleq \text{card}(\text{pos}(T))$ la taille de T pour cette représentation.

Une seconde représentation est celle des graphes dirigés acycliques (pointés), c'est-à-dire des arbres avec partage total (figure 2.2). Pour une signature et un ensemble de variable finis, une notion de taille acceptable pour cette représentation est le nombre de noeuds et de feuilles dans le graphe global, ce qui correspond au

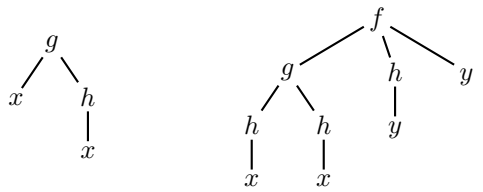

FIG. 2.1 – Les termes $g(x, h(x))$ et $f(g(h(x), h(x)), h(y), y)$ sous forme d'arbres sans partage

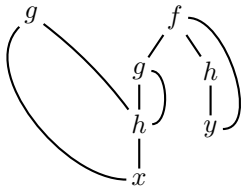

FIG. 2.2 – Les termes $g(x, h(x))$ et $f(g(h(x), h(x)), h(y), y)$ sous forme d'arbres avec partage total

nombre de sous-termes distincts dans le problème en entrée. Étant donné un terme T (ou plus généralement un n-uplet ou un ensemble de termes), on note $|T|_{\text{st}}$ le nombre de sous-termes distincts de T :

$$|T|_{\text{st}} \triangleq \text{card}(\text{st}(T))$$

Dans le cas d'une signature infinie ou d'un ensemble de variables infini (comme nous le supposons dans le reste du document), les notions précédentes ne sont pas rigoureusement adaptées à leur codage respectif. En effet, celles-ci ne prennent pas en compte la taille des noms de variables ni de symboles de fonction. Toutefois dans la suite nous nous conformons à l'usage répandu et ignorons la taille des noms de symboles dans les analyses de complexité. Ceci est généralement justifié par le fait que les problèmes considérés ne distinguent qu'un nombre fini de classes de symboles — les symboles d'une même classe pouvant être renommés arbitrairement.

2.5 Remplacements modulo E et symboles libres

Nous présentons maintenant une caractérisation de la notion de symboles libres en terme de remplacements modulo E.

Soit E une théorie équationnelle sur $\mathcal{F}[\mathcal{X}]$. Nous appelons *remplacement élémentaire modulo* E une expression

$$\rho = \{S \mapsto_E S'\}$$

où $S, S' \in \mathcal{F}[\mathcal{X}]$ sont de même sorte et $S = h(U_1, \ldots, U_m)$ n'est pas une variable. L'application d'un remplacement élémentaire $\rho = \{h(U_1, \ldots, U_m) \mapsto_E S'\}$ sur un

terme T, notée $T\rho$, est définie inductivement par

$$\begin{aligned} x\rho &= x \\ f(T_1,\dots,T_k) &= \begin{cases} S' & \text{si } f=h, k=m \text{ et } \forall i, T_i =_E U_i \\ f(T_1\rho,\dots,T_k\rho) & \text{sinon.} \end{cases} \end{aligned}$$

Plus généralement, un *remplacement modulo E* est une expression

$$\rho = \{S_1 \mapsto_E S'_1, \dots, S_n \mapsto_E S'_n\}$$

où chaque $S_i = h_i(U^i_1,\dots,U^i_{m_i})$ est un terme non variable de même sorte que S'_i, et pour tout $i \neq j$, $h_i = h_j$ implique l'existence d'un indice $1 \leq p \leq m_i$ tel que $U^i_p \neq_E U^j_p$. L'application d'un remplacement à un terme généralise celle des remplacements élémentaires de la manière attendue. (La seconde condition assure la non-ambiguïté de l'application.)

Lorsque $=_E$ est l'égalité ou que les termes S_i sont des constantes, on retrouve la notion usuelle de remplacement (pour une stratégie en largeur d'abord). Dans ce cas, on écrit simplement $\rho = \{S_1 \mapsto S'_1, \dots, S_n \mapsto S'_n\}$.

Exemple 2.1. À titre d'exemple, considérons la signature non sortée $\mathcal{F}$ comprenant un symbole binaire infixe $\oplus$ et un symbole binaire f, munie de la théorie équationnelle E du OU exclusif pour $\oplus$, c'est-à-dire engendrée par

$$\begin{aligned} x \oplus y &\doteq y \oplus x & \qquad x \oplus (y \oplus z) &\doteq (x \oplus y) \oplus z \\ x \oplus 0 &\doteq x & \qquad x \oplus x &\doteq 0 \end{aligned}$$

Soit $\rho = \{\mathsf{f}(x,y) \mapsto_E x\}$. On a par exemple :

$$\begin{aligned} \mathsf{f}(\mathsf{f}(x \oplus 0,\, y),\, 0 \oplus \mathsf{f}(x,\, y))\rho &=_E \mathsf{f}(x,\, 0 \oplus x) \\ \mathsf{f}(\mathsf{f}(x \oplus 0,\, y) \oplus y,\, z)\rho &=_E \mathsf{f}(x \oplus y,\, z) \end{aligned}$$

Comme le montre le premier exemple ci-dessus, en général, tous les termes égaux modulo E à un élément S_i du domaine de ρ ne sont pas nécessairement remplacés : intuitivement E ne s'applique qu'en dessous du symbole de tête de S_i. Cette particularité s'avère essentielle pour les utilisations de cette notion. Notamment, nous pouvons maintenant caractériser les symboles libres de manière syntaxique *et* sémantique.

Théorème et définition 2.2 (Symboles libres)**.** *Soit $h \in \mathcal{F}$. Les conditions suivantes sont équivalentes :*

(1) Il existe un ensemble d'équations $E_0 \subseteq E$ générant E tel que pour toute équation $(T \doteq S) \in E_0$, h n'apparaît pas dans T ni S.

(2) Pour tous termes T, S et $S_1 = h(U_1,\dots,U_m)$, et toute variable x de même sorte que S_1, $T =_E S$ implique $T\{S_1 \mapsto_E x\} =_E S\{S_1 \mapsto_E x\}$.

Un symbole de fonction h est dit libre *dans E s'il vérifie l'une des conditions ci-dessus.*

Démonstration. Montrons que (1) ⇒ (2). Soit $\mathcal{R} = \mathcal{R}(E_0)$ le système de règles associé à E_0. Par récurrence sur longueur de la dérivation $T =_{\mathcal{R}} S$, il suffit de traiter le cas où $T \to_{\mathcal{R}} S$ (par exemple).

Soit $l \to r \in \mathcal{R}$ (*i.e.* $l \doteq r \in E_0$), $p \in \mathrm{pos}(T)$ et σ tels que $T|_p = l\sigma$ et $S = T[r\sigma]_p$. Si p est une position strictement en dessous d'une position de remplacement dans T, alors par définition des remplacements modulo E, $T\rho = S\rho$.

Sinon, le remplacement ρ ne s'applique pas en p ni au-dessus de p dans T. On a alors

$$(T\rho)|_p = (T|p)\rho = (l\sigma)\rho = l(\sigma\rho)$$

où $\sigma\rho$ désigne l'application de ρ à σ ($\forall x \in \mathcal{X}, x(\sigma\rho) \stackrel{\triangle}{=} (x\sigma)\rho$), la dernière égalité étant due au fait que h n'apparaît pas dans l.

De même, $S\rho = (T\rho)[(r\sigma)\rho]_p = (T\rho)[r(\sigma\rho)]$. Par conséquent, $T\rho =_E S\rho$.

(2) $\Rightarrow$ (1). Pour chaque équation $T \doteq S \in E$, on définit une équation $\alpha(T \doteq S) \in E$ par induction sur le nombre de symboles h dans T et S :

- Si h n'apparaît pas, on prend $\alpha(T \doteq S) \stackrel{\triangle}{=} (T \doteq S)$.
- Sinon, soit $S_1 = h(U_1, \ldots, U_m)$ un sous-terme de T ou S, et x une variable n'apparaissant pas dans T et S. Sachant que l'équation $T\{S_1 \mapsto_E x\} \doteq S\{S_1 \mapsto_E x\}$) est bien dans E par (1) et qu'elle comporte strictement moins d'occurrences de h que $T \doteq S$, on peut définir

$$\alpha(T \doteq S) \stackrel{\triangle}{=} \alpha(T\{S_1 \mapsto_E x\} \doteq S\{S_1 \mapsto_E x\})$$

Soit $E_0 = \{\alpha(T \doteq S) \mid T \doteq S \in E\}$. Comme chaque $T \doteq S$ est une instance de $\alpha(T \doteq S)$, E_0 génère bien E. Or, par construction, h n'apparaît pas dans E_0. □

En utilisant la caractérisation précédente, nous pouvons maintenant établir deux propriétés attendues des symboles libres.

Corollaire 2.3. *Soit h un symbole libre dans E. Soit $h(T_1, \ldots, T_n)$ et $h(T'_1, \ldots, T'_n)$ deux termes d'une même sorte non dégénérée τ. On a l'équivalence suivante :*

$$h(T_1, \ldots, T_n) =_E h(T'_1, \ldots, T'_n) \quad \Leftrightarrow \quad \forall i,\ T_i =_E T'_i$$

Démonstration. L'implication $\Leftarrow$ est évidente. Pour la réciproque $\Rightarrow$, supposons par contradiction qu'il existe i_0 tel que $T_{i_0} \neq_E T'_{i_0}$ et que $h(T_1, \ldots, T_n) =_E h(T'_1, \ldots, T'_n)$. Soient x et x' deux variables fraîches distinctes de sorte τ. Appliquons le point (2) de la caractérisation précédente à cette égalité successivement avec $\rho_1 = \{h(T_1, \ldots, T_n) \mapsto_E x\}$ et $\rho_2 = \{h(T'_1\rho_1, \ldots, T'_n\rho_1) \mapsto_E x'\}$. On obtient $x =_E x'$, si bien que la sorte τ est dégénérée ; contradiction. □

Corollaire 2.4. *Soit h un symbole libre dans E. Soit $T = h(T_1, \ldots, T_n)$ et T' deux termes d'une même sorte non dégénérée τ. Si h n'apparaît pas dans T' alors $T \neq_E T'$.*

Démonstration. Supposons par contradiction que $T =_E T'$. Soit x une variable fraîche de sorte τ. Par le théorème 2.2, on a $x =_E T'\{T \mapsto_E x\}$. Or, comme h n'apparaît pas dans T', $T'\{T \mapsto_E x\} = T'$. Sachant que $x \notin \mathrm{var}(T')$, l'équation $x =_E T'$ implique que τ est dégénérée ; contradiction. □

Pour finir, nous décrivons un renforcement du théorème 2.2 dans le cas d'un ensemble de symboles libres et d'un remplacement arbitraire.

Proposition 2.5. *Soit $\rho = \{S_1 \mapsto_E S'_1, \ldots, S_n \mapsto_E S'_n\}$ un remplacement modulo E tel que tous les symboles de tête des S_i soient libres dans E. Alors pour tous termes T et S, $T =_E S$ implique $T\rho =_E S\rho$.*

Démonstration. La démonstration est identique à la preuve de (1) ⇒ (2) du théorème 2.2. □

Un cas particulier des trois énoncés précédents — que nous utiliserons abondamment dans les chapitres suivants — est celui des constantes libres. Le premier corollaire 2.3 signifie que si n et n' sont des constantes libres de sorte non dégénérée, alors $n =_E n'$ ssi $n = n'$. Le second corollaire 2.4 implique que si n est une constante libre de sorte non dégénérée, alors pour tout terme T ne contenant pas n, on a $n \neq_E T$. Enfin, la proposition 2.5 signifie que E est stable par remplacement (simultané) de constantes libres par des termes arbitraires.

Finalement, notons que le point (2) de la définition des symboles libres (théorème 2.2) peut être vu comme une propriété d'interpolation : lorsque $x \notin \mathrm{var}(T, S)$, prouver $T =_E S$ équivaut à prouver $T\{S_1 \mapsto x\} =_E S\{S_1 \mapsto x\}$. Nous proposons une généralisation de cette propriété à des symboles de certaines signatures dans le chapitre 6 (lemme 6.3).

2.6 Présentation canonique d'une théorie convergente

Appelons *théorie convergente* une théorie équationnelle engendrée par un système de réécriture convergent. Parmi les systèmes générant une théorie E, il est classique de distinguer les systèmes *interréduits* et *canoniques* au sens suivant.

Définition 2.1. Un système de réécriture $\mathcal{R}$ est *interréduit* lorsque pour toute règle $l \to r$ dans $\mathcal{R}$, r est $\mathcal{R}$-réduit et l est $(\mathcal{R} - \{l \to r\})$-réduit. Un système convergent et interréduit est dit *canonique*.

Il est bien connu que toute théorie convergente est présentable sous forme d'un système canonique. Nous utilisons plus loin la version suivante, plus précise, de cette propriété.

Proposition 2.6. *Soit $\mathcal{R}$ un système de réécriture convergent. Le système suivant $\mathcal{R}'$ est convergent, interréduit et engendre la même théorie équationnelle que $\mathcal{R}$:*

$$\mathcal{R}' = \{l \to (r\downarrow_{\mathcal{R}}) \mid (l \to r) \in \mathcal{R} \text{ et } l \text{ est } (\mathcal{R} - \{l \to r\})\text{-réduit } \}$$

Démonstration. Clairement, $\mathcal{R}'$ est interréduit et $(\to_{\mathcal{R}'}) \subseteq (\to^+_{\mathcal{R}})$. En particulier, $\mathcal{R}'$ termine. Montrons que pour tout T, on a $T \to^*_{\mathcal{R}'} T\downarrow_{\mathcal{R}}$, par induction sur $(T, \to_{\mathcal{R}})$ ($\mathcal{R}$ étant terminant). Sachant que $(\to_{\mathcal{R}'}) \subseteq (\to^+_{\mathcal{R}})$, ceci impliquera en effet à la fois la confluence de $\mathcal{R}'$ et $(=_{\mathcal{R}}) = (=_{\mathcal{R}'})$.

Le cas $T = T\downarrow_{\mathcal{R}}$ est trivial. Supposons $T \to_{\mathcal{R}} T' \to^*_{\mathcal{R}} T\downarrow_{\mathcal{R}}$. Soit $(l \to r) \in \mathcal{R}$, σ et p tels que $T|_p = l\sigma$ et $T' = T[r\sigma]_p$.

Si l est $(\mathcal{R} - \{l \to r\})$-réduit, en particulier $l \to (r\downarrow_{\mathcal{R}})$ est dans $\mathcal{R}'$, et $l \to^*_{\mathcal{R}'} (r\downarrow_{\mathcal{R}})$.

Sinon, il existe $(l' \to r') \in \mathcal{R} - \{l \to r\}$, σ' et q tels que $l|_q = l'\sigma'$. Alors, par confluence de $\mathcal{R}$, $l \to^*_{\mathcal{R}} (r\downarrow_{\mathcal{R}})$ et $l \to_{\mathcal{R}} l[r'\sigma']_q$ impliquent $l[r'\sigma']_q \to^*_{\mathcal{R}} (r\downarrow_{\mathcal{R}})$. Or,

$\mathcal{R}$ étant terminant, pour tout $l[r'\sigma']_q \to^*_{\mathcal{R}} U$, U ne peut pas être réductible par $l \to r$. On a donc également $l \to^*_{\mathcal{R}'} (r\downarrow_{\mathcal{R}})$.

Sachant que $T \to_{\mathcal{R}} T' \to^*_{\mathcal{R}} T\downarrow_{\mathcal{R}}$ et $T = T[l\sigma]_p \to^*_{\mathcal{R}} T[(r\downarrow_{\mathcal{R}})\sigma]_p$, par confluence de $\mathcal{R}$, on a $T[(r\downarrow_{\mathcal{R}})\sigma]_p \to^*_{\mathcal{R}} T\downarrow_{\mathcal{R}}$. Or, $T \to^+_{\mathcal{R}} T[(r\downarrow_{\mathcal{R}})\sigma]_p$, d'où finalement, par hypothèse d'induction

$$T \to^*_{\mathcal{R}'} T[(r\downarrow_{\mathcal{R}})\sigma]_p \to^*_{\mathcal{R}'} T\downarrow_{\mathcal{R}} .$$ □

Enfin, le résultat technique suivant montre qu'un système de réécriture canonique ne comporte pas de symboles libres dans la théorie équationnelle associée. Grâce à la proposition précédente, ceci nous dispense de distinguer les notions de symbole libre dans un système de réécriture et dans une théorie équationnelle.

Proposition 2.7. *Soit E une théorie équationnelle engendrée par un système de réécriture canonique $\mathcal{R}$. Alors, les règles de $\mathcal{R}$ ne contiennent aucun symbole libre dans E.*

Démonstration. Par contradiction, supposons qu'il existe une règle $l \to r$ dans $\mathcal{R}$ contenant un sous-terme $T_0 = \mathsf{h}(U_1, \ldots, U_n)$ tel que h est libre dans E.

Soit x une variable fraîche de même sorte que T_0. Posons $l' = l\{T_0 \mapsto_E x\}$, $r' = r\{T_0 \mapsto_E x\}$ et $\mu = \{x \mapsto T_0\}$.

Par définition des symboles libres, on a $l' =_E r'$. De plus, $\mathcal{R}$ étant interréduit, tous les sous-termes de r et tous les sous-termes stricts de l sont $\mathcal{R}$-réduits.

Si $l' = l\{T_0 \mapsto_E x\} = x$ alors $x =_E r'$ implique que la sorte τ est dégénérée, ce qui contredit la proposition 2.1. Par conséquent, le remplacement précédent opère uniquement dans les sous-termes de r et dans les sous-termes stricts de l. Il est donc syntaxique : $(l' \to r') = (l \to r)\{T_0 \mapsto x\}$, de sorte que $l = l'\mu$ et $r = r'\mu$.

Or, par confluence $l' =_E r'$ implique $l' \to^*_{\mathcal{R}} \leftarrow^*_{\mathcal{R}} r'$. $r = r'\mu$ étant $\mathcal{R}$-réduit, r' l'est également et $l' \to^*_{\mathcal{R}} r'$. Mais, $\mathcal{R}$ étant terminant, pour tout $l' \to^*_{\mathcal{R}} U$, U ne contient pas d'instance de l', ni donc de $l = l'\mu$. On en déduit $l' \to^*_{\mathcal{R}-\{l\to r\}} r'$ puis $l \to^*_{\mathcal{R}-\{l\to r\}} r$, ce qui contredit le fait que $\mathcal{R}$ est interréduit. □

2.7 Théories sous-terme-convergentes

Nous terminons ce chapitre par la définition d'une classe de théories équationnelles permettant de modéliser de nombreuses primitives cryptographiques classiques, notamment les chiffrements de type symétrique, asymétrique, probabiliste ou déterministe, les signatures, les fonctions de hachage [AF01, Bla04, AC04, CDE04]. Nous proposons des exemples de telles théories dans les chapitres 3 et 6. Le chapitre 4, quant à lui, est dédié à la résolution d'une classe de système de contraintes paramétrés par une théorie sous-terme-convergente.

Définition 2.2. Un système de réécriture $\mathcal{R}$ a la propriété *sous-terme* lorsque pour toute règle $l \to r \in \mathcal{R}$, r est ou bien un sous-terme strict de l, ou bien un terme clos en forme $\mathcal{R}$-normale.

On appelle *sous-terme-convergent* un système de réécriture convergent ayant la propriété sous-terme.

Par extension, une théorie équationnelle E est *sous-terme-convergente* ssi elle est engendrée par un système de réécriture sous-terme-convergent.

Cette définition s'inspire de celle proposée par M. Abadi et V. Cortier [AC04]. La différence entre les deux définitions réside dans le fait que, dans le cas où r n'est pas un sous-terme de l, nous autorisons ici un terme clos, $\mathcal{R}$-normal r arbitraire plutôt qu'une constante. Par rapport à la définition donnée dans un premier article [Bau05], nous supprimons également une restriction mineure sur ce même r (à savoir : les symboles de r sont pris dans un sous-ensemble fixé $\mathcal{F}_{\text{pub}} \subseteq \mathcal{F}$).

Une autre classe de théories équationnelles, appelées *publiques-effondrantes* (*public collapsing*) est proposée par S. Delaune et F. Jacquemard [DJ04a]. Étant donné un sous-ensemble de symboles $\mathcal{F}_{\text{pub}} \subseteq \mathcal{F}$, dits *publics*, un système de réécriture $\mathcal{R}$ est *public-effondrant* ssi pour toute règle $l \to r \in \mathcal{R}$, les conditions suivantes sont vérifiées :

1. $r \neq l$ et au choix $r \in \text{var}(l)$ ou $r \in \mathcal{F}_{\text{pub}}[\emptyset]{\downarrow_{\mathcal{R}}}$ est un terme public, clos et $\mathcal{R}$-réduit ;
2. si $l = f(l_1, \ldots, l_n)$ et $f \in \mathcal{F}_{\text{pub}}$, alors pour tout sous-terme strict de l de la forme $g(t_1, \ldots, t_m)$ où $g \in \mathcal{F}_{\text{pub}}$, on a ou bien $g(t_1, \ldots, t_m) \in \mathcal{F}_{\text{pub}}[\emptyset]{\downarrow_{\mathcal{R}}}$ ou il existe j tel que $t_j = r$.

Enfin, une théorie est *publique-effondrante* ssi elle est engendrée par un système de réécriture convergent et public-effondrant.

Ces définitions impliquent que tout système de réécriture public-effondrant a en particulier la propriété sous-terme, ou encore que toute théorie publique-effondrante est sous-terme-convergente.

Inversement, certaines systèmes de réécriture sous-terme-convergents ne sont pas publics-effondrants. C'est le cas par exemple des modélisations suivantes, respectivement du chiffrement asymétrique[2], de la signature et d'un symbole idempotent h :

$$\begin{aligned}
\mathcal{R}_1 &= \{\mathsf{pdec}(\mathsf{penc}(x, \mathsf{pub}(y)), y, z) \to x\} \\
\mathcal{R}_2 &= \{\mathsf{check}(\mathsf{sig}(x, y, z), \mathsf{pub}(y)) \to \mathsf{ok}\} \\
\mathcal{R}_3 &= \{\mathsf{h}(\mathsf{h}(x)) \to \mathsf{h}(x)\}
\end{aligned}$$

les symboles mentionnés étant publics.

Nous établissons maintenant plusieurs propriétés utiles des théories ou systèmes sous-terme-convergents. La première concerne leur terminaison.

Proposition 2.8. *Tout système de réécriture $\mathcal{R}$ ayant la propriété sous-terme est terminant. En particulier, un tel système est convergent ss'il est localement confluent.*

Démonstration. Pour tout terme T, on note $\mu(T)$ le nombre de positions $p \in \text{pos}(T)$ telles $T|_p$ n'est pas en forme $\mathcal{R}$-normale. Montrons que $T \to_{\mathcal{R}} T'$ implique $\mu(T) > \mu(T')$.

En effet, soit $p \in \text{pos}(T)$, $l \to r \in \mathcal{R}$ et σ tels que $T|_p = l\sigma$ et $T' = T[r\sigma]_p$. Si $r\sigma = r \in \mathcal{F}[\emptyset]$ est $\mathcal{R}$-normal, alors $\mu(T') \leq \mu(T) - 1$. Sinon, $r\sigma$ étant un sous-terme strict de $l\sigma$, on a également $\mu(T') \leq \mu(T) - 1$. □

[2]Dans le cas du chiffrement asymétrique, S. Delaune et F. Jacquemard [DJ04a] proposent une modélisation différente à l'aide d'un symbole privé involutif $(\_)^{-1}$.

Notons que la propriété sous-terme n'implique pas la confluence, comme le montre l'exemple suivant :

$$\mathcal{R}_4 = \{\mathsf{f}(\mathsf{g}(x)) \to \mathsf{a}, \quad \mathsf{g}(\mathsf{f}(x)) \to \mathsf{b}\}$$

vérifie $\mathsf{f}(\mathsf{g}(\mathsf{f}(\mathsf{c}))) \to_{\mathcal{R}_4} \mathsf{a}$ et $\mathsf{f}(\mathsf{g}(\mathsf{f}(\mathsf{c}))) \to_{\mathcal{R}_4} \mathsf{f}(\mathsf{b})$.

Enfin, les propositions 2.6 et 2.7 de la section précédente s'appliquent aux théories sous-terme-convergentes de la manière suivante.

Proposition 2.9. *Soit E une théorie équationnelle engendrée par un système de réécriture sous-terme-convergent $\mathcal{R}$. Le système de réécriture*

$$\begin{aligned} \mathcal{R}' &= \{l \to (r\downarrow_{\mathcal{R}}) \mid (l \to r) \in \mathcal{R} \textit{ et } l \textit{ est } (\mathcal{R} - \{l \to r\})\textit{-réduit } \} \\ &= \{(l \to r) \in \mathcal{R} \mid l \textit{ est } (\mathcal{R} - \{l \to r\})\textit{-réduit } \} \end{aligned}$$

est sous-terme-convergent, interréduit et engendre E. De plus, $\mathcal{R}'$ ne contient aucun symbole libre de E.

Démonstration. D'après les propositions 2.6 et 2.7, il reste uniquement à vérifier que $\mathcal{R}'$ a la propriété sous-terme. Or, pour tout $l \to r$ dans $\mathcal{R}$, si r n'est pas $\mathcal{R}$-réduit, alors r est un sous-terme strict de l, qui n'est donc pas $(\mathcal{R} - \{l \to r\})$-réduit. □

Première partie

Sécurité logique des protocoles cryptographiques

Chapitre 3

Spécification de protocoles cryptographiques dans une algèbre de processus

Sommaire

Une manière répandue de décrire un protocole cryptographique — utilisée par exemple dans le recueil de J. Clark et J. Jacob [CJ97] — est de fournir la séquence des messages échangés lors d'une session réussie du protocole. Ainsi, le protocole Handshake [GLNS93, DJ04b] s'écrit par exemple :

$$\begin{array}{lll} 0. & A \to B : & \{N\}_{k_{AB}} \\ 1. & B \to A : & \{f(N)\}_{k_{AB}} \end{array}$$

Le but de ce protocole très simple est d'authentifier B du point de vue de A, sachant que A et B partage un certain secret k_{AB}. Pour ce faire,

(i) A envoie à B un nombre aléatoire frais chiffré par la clé secrète k_{AB} ;

(ii) B récupère ce nombre en déchiffrant le message, y applique une certaine fonction (par exemple $f(N) = N + 1$), et renvoie le résultat à A, également chiffré par k_{AB} ;

(iii) finalement A vérifie la validité du dernier message, et en conclut (ou non) la présence de B.

Comme on peut le voir, la séquence des messages du protocole donne peu d'indication sur les calculs que doit effectuer chaque agent pour s'y conformer. C'est pourquoi la séquence des messages s'accompagne généralement d'une explication littérale du protocole — comme nous venons de le faire.

Notons que l'étape (iii), implicite dans la séquence des messages, peut être réalisée de deux manières différentes : déchiffrer le message et comparer le résultat à $f(N)$, ou bien calculer le chiffré de $f(N)$ et le comparer au message reçu (cette dernière méthode ne fonctionnant que pour un chiffrement déterministe).

En tant que description du protocole, la séquence des messages est donc ambiguë. Pire, une séquence de messages peut s'avérer impossible à réaliser. Par exemple, si la fonction f n'est pas inversible (en pratique), la séquence suivante n'est pas réalisable par B :

$$\begin{array}{lll} 0. & A \to B : & \{f(N)\}_{k_{AB}} \\ 1. & B \to A : & \{N\}_{k_{AB}} \end{array}$$

Enfin, la propriété de sécurité voulue du protocole reste à formaliser. Comme nous le verrons, plusieurs notions d'authentification sont d'ailleurs envisageables [Low97].

Pour toutes ces raisons, la vérification logique d'un protocole cryptographique nécessite au préalable sa modélisation dans un formalisme adapté.

De nombreux modèles ont été proposés pour réaliser cette tâche (voir le chapitre 1). Toutefois, dans la lignée du spi-calcul de M. Abadi et d'A. Gordon [AG97], un certain nombre de modèles récents [AG98, BdNP99, DSV03, AF01, Bor01, Cor02, Bla02, Hüt00, BDNN02, AB05a, BAF05, BN05, BB05, ZD06] reposent sur des langages de programmation concurrents, spécialisés aux protocoles cryptographiques. L'intérêt de ces langages est au moins double :

– d'une part, les protocoles, une fois modélisés comme des programmes du langage (également appelés *processus*), sont des objets que l'on peut exécuter : en termes savants, ils bénéficient de la sémantique opérationnelle du langage ; ceci contribue à clarifier la modélisation des protocoles ainsi que des propriétés de sécurité désirées dans la mesure où celles-ci s'expriment le plus souvent comme des garanties sur l'exécution du protocole ;

– d'autre part, les notions d'équivalence observationnelle héritées de la théorie des langages de programmation permettent de modéliser des propriétés fines de sécurité reposant sur l'indistinguabilité de deux scénarios du point de vue de l'intrus [AG97, AG98, BdNP99, DSV03, AF01, Cor02, Hüt00, BAF05]. Parmi ces propriétés on compte notamment la résistance aux attaques par dictionnaire qui bénéficie d'une définition élégante en terme d'équivalence de processus [CDE04, BAF05].

Ce chapitre a pour but de montrer comment spécifier et étudier une classe significative de protocoles cryptographiques et de propriétés de sécurité au moyen d'un tel langage. Dans un premier temps (section 3.1), nous décrivons la syntaxe et la sémantique opérationnelle du langage utilisé. Nous détaillons ensuite dans la section 3.2 comment modéliser certaines propriétés de sécurité usuelles, comme le secret et l'authentification, ou moins habituelles, comme l'absence d'attaques par dictionnaire et le secret par anticipation (*forward secrecy*).

Les deux sections suivantes sont consacrées à la simplification de l'expression des propriétés de sécurité au moyen d'une sémantique dite ouverte (section 3.3) puis d'une sémantique symbolique (section 3.4). Enfin, nous utilisons ces résultats dans la section 3.5 pour étudier ces propriétés d'un point de vue algorithmique. Notamment, pour la plupart d'entre elles, nous réduisons le problème de la sécurité d'un protocole fini (*i.e.* pour un nombre borné de sessions) à une classe de problèmes purement équationnels à base d'E-unification du second ordre. L'étude algorithmique de ces problèmes fait l'objet du chapitre 4.

3.1 Syntaxe et sémantique opérationnelle

3.1.1 Syntaxe

On suppose fixés un ensemble de sortes $\mathcal{T}$, un ensemble infini de variables $\mathcal{X}^1$ dont les éléments sont notés $x, y \ldots$, ainsi qu'un ensemble de symboles de fonction $\mathcal{F} = \mathcal{F}_{\text{pub}} \uplus \mathcal{F}_{\text{priv}} \uplus \mathcal{N}$ partitionné en symboles *publics* $\mathcal{F}_{\text{pub}}$, *privés* $\mathcal{F}_{\text{priv}}$, et en constantes particulières appelées *noms* $\mathcal{N}$.

L'ensemble des termes construits sur la signature $(\mathcal{F}, \mathcal{X}^1)$, noté $\mathcal{F}[\mathcal{X}^1]$, est muni d'une théorie équationnelle E. On suppose qu'il existe un nombre infini de noms de chaque sorte dans $\mathcal{N}$ et que les noms sont libres dans E. Autrement dit, par la proposition 2.5, $=_E$ est stable par remplacement de noms par des termes quelconques (de sorte appropriée). On appelle *renommage de nom* (ou simplement *renommage*) un remplacement bijectif ρ entre deux sous-ensembles finis de noms.

Pour des raisons techniques, on suppose que $\mathcal{F}_{\text{pub}}$ contient également un nombre infini de constantes libres de chaque sorte. Les symboles publics $\mathcal{F}_{\text{pub}}$, par opposition aux symboles privés $\mathcal{F}_{\text{priv}}$ et aux noms $\mathcal{N}$, représentent des opérations et des données toujours accessibles à l'attaquant.

Dans la suite, les lettres $s, t, u \ldots$ désignent des éléments de $\mathcal{F}[\mathcal{X}^1]$, tandis que les noms sont notés $n, a, b, c \ldots$

Considérons le langage de *processus* défini par la grammaire suivante :

P, Q	::=		processus
	|	0	processus vide
	|	$P \mid Q$	composition parallèle
	|	$!\,P$	réplication
	|	$u(x).\,P$	réception sur le canal u
	|	$\bar{u}\langle v\rangle.\,P$	émission de v sur le canal u
	|	if $u = v$ then P else Q	test d'égalité (u et v de même sorte)
	|	new $x.\,P$	création de nom
	|	wait $.P$	attente jusqu'à la phase suivante

Pour l'essentiel, ce langage reprend les constructions du pi-calcul appliqué [AF01], lui-même conçu comme une extension du pi-calcul de R. Milner [Mil93] par une algèbre de terme munie d'une théorie équationnelle. Ainsi, l'on dispose des opérations classiques de composition parallèle, de réplication, de réception, d'émission d'un message, et de création de noms. Cette dernière opération représente la génération d'une valeur fraîche, c'est-à-dire d'un point de vue cryptographique, le tirage d'un nombre « parfaitement » aléatoire.

L'instruction wait modélise l'attente de la *phase* suivante d'un processus. Par exemple, si le processus Q ne contient pas de wait, $Q \mid \text{wait}\,.P$ commence par exécuter un certain nombre d'étape de Q, puis supprime le reste du processus Q et exécute le processus P seul. Une extension comparable du pi-calcul appliqué par des phases est introduite dans [BAF05, ZD06]. Un processus est dit *en une phase* s'il ne contient pas d'instruction wait. Plus généralement, le *nombre de phases* d'un processus P est le nombre maximal d'instructions wait imbriquées dans P plus un :

$$\begin{aligned}
\mathrm{nph}(0) &= 1\\
\mathrm{nph}(u(x).C) &= \mathrm{nph}(\mathrm{new}\,x.C) = \mathrm{nph}(!\,C)\\
&= \mathrm{nph}(\bar{u}\langle v\rangle.C) = \mathrm{nph}(C)\\
\mathrm{nph}(P|Q) &= \mathrm{nph}(\text{if } u = v \text{ then } P \text{ else } Q)\\
&= \max(\mathrm{nph}(P), \mathrm{nph}(Q))\\
\mathrm{nph}(\text{wait}\,.C) &= 1 + \mathrm{nph}(C)
\end{aligned}$$

On appelle *processus positifs* la classe des processus dans lesquels les tests d'égalité sont uniquement de la forme : if $u = v$ then P else 0.

Les notations names(t) et names(P) désignent l'ensemble des noms contenus respectivement dans t et P. De même, var(P) représente l'ensemble des variables libres de P : $\mathrm{var}(u(x).P) = \mathrm{var}(\mathrm{new}\,x.\,P) \triangleq \mathrm{var}(P) - \{x\}$, var($P$) étant défini de la manière évidente dans tous les autres cas . Les variables non libres de P sont dites *liées*. Un processus P est *clos* s'il ne contient pas de variables libres : $\mathrm{var}(P) = \emptyset$.

Comme il est d'usage, les processus sont considérés modulo renommage des variables liées. Les substitutions (éventuellement partielles) σ sont appliquées aux processus de manière à protéger les variables liées, par exemple : $(\mathrm{new}\,x.P)\sigma = \mathrm{new}\,x.P\sigma$ si $x \notin \mathrm{supp}(\sigma) \cup \mathrm{var}(\mathrm{supp}(\sigma)\sigma)$.

Un terme t est dit *public* s'il appartient à $\mathcal{F}_{\mathrm{pub}}[\mathcal{X}^1]$, c'est-à-dire si t ne contient pas de noms ni de symboles privés. Un processus P (et plus généralement un ensemble de processus) est *public* ssi tous les termes apparaissant dans P sont publics.

$$\mathcal{E};\ \{0\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{(nil)}$$

$$\mathcal{E};\ \{P \mid Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \{P,\ Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{(par)}$$

$$\mathcal{E};\ \{!\,P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \{P,\ !\,P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{(repl)}$$

$$\mathcal{E};\ \{u(x).\,P,\ \bar{v}\langle t\rangle.\,Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \{P\{x \mapsto t\},\ Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{(comm)}$$
$$\text{si } \begin{cases} \mathrm{sort}(x) = \mathrm{sort}(t) \\ u =_E v \end{cases}$$

$$\mathcal{E};\ \{\text{if } u = v \text{ then } P \text{ else } Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \{P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{si } u =_E v \qquad \text{(test-1)}$$

$$\mathcal{E};\ \{\text{if } u = v \text{ then } P \text{ else } Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \{Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{si } u \neq_E v \qquad \text{(test-2)}$$

$$\mathcal{E};\ \{\text{new } x.P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \{n\} \cup \mathcal{E};\ \{P\{x \mapsto n\}\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{(new)}$$
$$\text{si } \begin{cases} n \in \mathcal{N} - \mathcal{E} \\ \mathrm{sort}(n) = \mathrm{sort}(x) \end{cases}$$

$$\mathcal{E};\ \{\text{wait }.P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \{P\} \uplus \mathcal{P}_{\mathrm{a}} \qquad \text{(wait)}$$

$$\mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \mathcal{P}_{\mathrm{a}};\ \emptyset \qquad \text{si } \mathcal{P}_{\mathrm{a}} \neq \emptyset \qquad \text{(next)}$$

Tab. 3.1 – Transitions entre configurations

3.1.2 Sémantique à petits pas

Nous décrivons maintenant l'exécution pas à pas des processus au moyen d'une relation de transition entre configurations.

Définition 3.1. Une *configuration* est un triplet $\mathcal{K} = \mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}}$ où
- $\mathcal{E}$ est un ensemble fini de noms,
- $\mathcal{P}_{\mathrm{e}}$ et $\mathcal{P}_{\mathrm{a}}$ sont deux multi-ensembles finis de processus clos vérifiant l'inclusion $\mathrm{names}(\mathcal{P}_{\mathrm{e}}, \mathcal{P}_{\mathrm{a}}) \subseteq \mathcal{E}$.

Les éléments de $\mathcal{P}_{\mathrm{e}}$ et $\mathcal{P}_{\mathrm{a}}$ sont appelés respectivement les *processus en cours d'exécution* et les *processus en attente* de $\mathcal{K}$. Les triplets $\mathcal{K} = \mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}}$ ne satisfaisant pas nécessairement la condition $\mathrm{names}(\mathcal{P}_{\mathrm{e}}, \mathcal{P}_{\mathrm{a}}) \subseteq \mathcal{E}$ sont appelés *pré-configurations*.

On définit la relation de *transition à petits pas* $\to$ sur les configurations par les règles de la table 3.1. Les règles (nil), (par) et (repl) correspondent respectivement à la terminaison d'un processus, la mise en parallèle et la réplication. La règle (comm) permet la communication d'un message t sur le canal $u =_E v$ pourvu que t soit de la même sorte que la variable x censée le recevoir. Notons que les canaux sont ici des termes quelconques. De plus, chaque canal permet d'envoyer n'importe quel type de donnée. Les règles (test-1) et (test-2) décrivent le comportement attendu des tests d'égalité. La règle (new) permet la création d'un nom frais n. Enfin, la règle

(wait) met en attente le processus P, tandis que (next) termine tous les processus en cours d'exécution et active les processus en attente.

Concernant la sémantique du wait, on remarque que le nombre d'exécutions de la règle (next) au cours d'une dérivation est borné, ce malgré la présence de l'opérateur de réplication. Plus précisément, pour tout $\mathcal{K} = \mathcal{E}; \mathcal{P}_\mathrm{e}; \mathcal{P}_\mathrm{a}$, notons

$$\mathrm{nph}(\mathcal{K}) \triangleq \max\{0\} \cup \{\mathrm{nph}(P), P \in \mathcal{P}_\mathrm{e}\} \cup \{\mathrm{nph}(P) + 1, P \in \mathcal{P}_\mathrm{a}\}$$

Pour tout $\mathcal{K} \to \mathcal{K}'$, on a $\mathrm{nph}(\mathcal{K}) \geq \mathrm{nph}(\mathcal{K}')$, l'inégalité étant stricte pour la règle (next).

Notons enfin que $\mathcal{E}; \mathcal{P}_\mathrm{e}; \mathcal{P}_\mathrm{a} \to^* \mathcal{E}'; \mathcal{P}'_\mathrm{e}; \mathcal{P}'_\mathrm{a}$ implique $\mathcal{E} \subseteq \mathcal{E}'$. Par la suite, on identifiera au besoin un processus P avec la configuration $\mathcal{K}(P) = \mathrm{names}(P); \{P\}; \emptyset$.

Étant donné deux pré-configurations $\mathcal{K} = \mathcal{E}; \mathcal{P}_\mathrm{e}; \mathcal{P}_\mathrm{a}$ et $\mathcal{K} = \mathcal{E}'; \mathcal{P}'_\mathrm{e}; \mathcal{P}'_\mathrm{a}$ telles que $\mathcal{E} \cap \mathcal{E}' = \emptyset$, la *composée* de $\mathcal{K}$ et de $\mathcal{K}'$ est définie par

$$\mathcal{K} \,|\, \mathcal{K}' \triangleq \mathcal{E} \uplus \mathcal{E}'; \mathcal{P}_\mathrm{e} \uplus \mathcal{P}'_\mathrm{e}; \mathcal{P}_\mathrm{a} \uplus \mathcal{P}'_\mathrm{a}.$$

Afin de faciliter la lecture des exemples, nous définissons les abréviations suivantes :

$$\begin{aligned}
u(x) &\triangleq u(x).0 \\
\overline{u}\langle v\rangle &\triangleq \overline{u}\langle v\rangle.0 \\
\mathrm{let}\, x = u \,\mathrm{in}\, P &\triangleq P\{x \mapsto u\} \\
\mathrm{new}\, x_1, \ldots, x_n \,\mathrm{in}\, P &\triangleq \mathrm{new}\, x_1 \,\mathrm{in} \ldots \mathrm{new}\, x_n \,\mathrm{in}\, P \\
\mathrm{if}\, u = v \,\mathrm{then}\, P &\triangleq \mathrm{if}\, u = v \,\mathrm{then}\, P \,\mathrm{else}\, 0 \\
\mathrm{if}\, u \neq v \,\mathrm{then}\, P \,\mathrm{else}\, Q &\triangleq \mathrm{if}\, u = v \,\mathrm{then}\, Q \,\mathrm{else}\, P \\
\mathrm{if}\, u = v \;\mathrm{and}\; u' = v' \,\mathrm{then}\, P \,\mathrm{else}\, Q &\triangleq \mathrm{if}\, u = v \,\mathrm{then} \\
&\qquad (\,\mathrm{if}\, u' = v' \,\mathrm{then}\, P \,\mathrm{else}\, Q\,) \\
&\qquad \mathrm{else}\, Q \\
\mathrm{if}\, u = v \;\mathrm{or}\; u' = v' \,\mathrm{then}\, P \,\mathrm{else}\, Q &\triangleq \mathrm{if}\, u = v \,\mathrm{then}\, P \,\mathrm{else} \\
&\qquad \mathrm{if}\, u' = v' \,\mathrm{then}\, P \,\mathrm{else}\, Q
\end{aligned}$$

Par extension, on définit de même l'abréviation : if *cond* then P(else Q), où *cond* représente une formule booléenne quelconque à base de tests d'égalité.

Enfin, on écrit $\overline{u}\langle\rangle.P$ pour l'envoi sur un canal u d'un message arbitraire, par exemple u lui-même. L'expression $u().P$ représente alors le processus $u(x).P$ où $x \notin \mathrm{var}(P)$ est une variable quelconque de même sorte que u.

Exemple 3.1. À titre d'illustration, considérons à nouveau le protocole Handshake :

$$\begin{array}{lll} 0. & A \to B : & \{N\}_{k_{AB}} \\ 1. & B \to A : & \{f(N)\}_{k_{AB}} \end{array}$$

Pour modéliser ce protocole (ainsi que les propriétés de sécurité décrites plus loin), on considère la signature non typée suivante :

$$\begin{aligned}
\mathcal{F} &= \mathcal{F}_\mathrm{pub} \uplus \mathcal{F}_\mathrm{priv} \uplus \mathcal{N} \\
\mathcal{F}_\mathrm{pub} &= \{\mathsf{enc}(2),\ \mathsf{dec}(2),\ \mathsf{f}(1),\ \mathsf{c},\ \mathsf{i},\ \mathsf{h}(3)\} \cup \mathcal{F}_c \\
\mathcal{F}_\mathrm{priv} &= \{\mathsf{k}(2)\}
\end{aligned}$$

où les chiffres entre parenthèses spécifient l'arité des symboles non constants, tandis que $\mathcal{F}_c$ et $\mathcal{N}$ désignent des ensembles de constantes, en nombre infini pour chaque sorte. On munit cette signature de la théorie équationnelle E_0 engendrée par les équations d'un chiffrement symétrique déterministe surjectif (ou *cipher*) :

$$\mathsf{dec}(\mathsf{enc}(x,y),y) \doteq x \qquad \mathsf{enc}(\mathsf{dec}(x,y),y) \doteq x$$

Les autres symboles de la signature sont donc libres. La constante c est utilisée comme canal de communication, et i comme nom de l'attaquant (*intruder*). Le symbole libre f modélise la fonction f, tandis que h sera utilisé plus bas dans l'énoncé de certaines propriétés de sécurité.

Les rôles A et de B du protocole sont alors modélisés par les processus suivants, paramétrés par les noms des agents ($a, b \in \mathcal{X}^1$) :

$$\begin{aligned} P_A(a,b) \quad &\triangleq \quad \mathrm{new}\, x.\ \overline{\mathsf{c}}\langle\, \mathsf{enc}(x, \mathsf{k}(a,b))\,\rangle.\ \mathsf{c}(y). \\ &\qquad\qquad \mathrm{if}\ \mathsf{dec}(y, \mathsf{k}(a,b)) = \mathsf{f}(x)\ \mathrm{then}\, 0\ (*\ \text{succès}\ *) \\ &\qquad\qquad \mathrm{else}\, 0\ (*\ \text{échec}\ *) \\ P_B(a,b) \quad &\triangleq \quad \mathsf{c}(z).\ \mathrm{let}\, x = \mathsf{dec}(z, \mathsf{k}(a,b))\ \mathrm{in}\ \overline{\mathsf{c}}\langle \mathsf{enc}(\mathsf{f}(x), \mathsf{k}(a,b))\rangle \end{aligned}$$

Enfin, le protocole complet est modélisé par le processus P_0 :

$$\begin{aligned} P_0 \quad \triangleq \quad & !\,\mathsf{c}(a).\ \mathsf{c}(b).\ P_A(a,b) \quad | \quad !\,\mathsf{c}(a).\ \mathsf{c}(b).\ P_B(a,b) \\ & \mid !\,\mathsf{c}(a).\ \overline{\mathsf{c}}\langle k(a,\mathsf{i})\rangle.\ \overline{\mathsf{c}}\langle k(\mathsf{i},a)\rangle \end{aligned}$$

La première ligne permet à l'intrus d'initier un nombre arbitraire de sessions entre les agents de son choix, tandis que la seconde lui donne la possibilité d'obtenir une clé partagée avec n'importe quel agent.

On peut d'ores et déjà vérifier que le protocole est bien exécutable. En effet, soit a et b deux constantes. Une session normale du protocole entre les agents a et b se déroule de la manière décrite par la table 3.2.

Remarque. À la différence du pi-calcul appliqué [AF01], nous avons choisi de définir la sémantique opérationnelle du langage sans utiliser d'équivalence structurelle, de façon comparable à X. Allamigeon et B. Blanchet [AB05b]. Ce choix est surtout guidé par les applications en vue, pour lesquelles on souhaite une sémantique opérationnelle aussi explicite que possible. Le prix à payer est que la relation de transition est définie non pas sur les processus (modulo équivalence structurelle) mais sur les configurations.

3.1.3 Expressivité du langage

Primitives cryptographiques. Le formalisme de la logique équationnelle permet de modéliser un large éventail de fonctions cryptographiques [DY83, AF01, CDE04, AC05, CDL06]. À titre d'illustration, nous présentons dans les tables 3.3 et 3.4 les modélisations possibles de différentes primitives sous forme de théorie équationnelle. (Tous les symboles mentionnés sont publics et construits sur une signature comprenant une unique sorte.)

Pour commencer, nous considérons dans la table 3.3 plusieurs primitives modélisées naturellement par des théories équationnelles sous-terme-convergentes (définition 2.2). Ces exemples s'inspirent de modélisations classiques [AF01, AC04, DJ04a, CDE04].

$\mathcal{K}(\, P_A(\mathsf{a},\mathsf{b}) \,|\, P_B(\mathsf{a},\mathsf{b}) \,)$

$$
\begin{array}{rl}
= & \emptyset;\ \{ \\
& \quad \mathrm{new}\, x.\ \overline{\mathsf{c}}\langle\, \mathsf{enc}(x, \mathsf{k}(\mathsf{a},\mathsf{b}))\,\rangle.\ \mathsf{c}(y). \\
& \qquad \mathrm{if}\ \mathsf{dec}(y, \mathsf{k}(\mathsf{a},\mathsf{b})) = \mathsf{f}(x)\ \mathrm{then}\, 0\ \text{(* succès *)} \\
& \qquad \mathrm{else}\, 0\ \text{(* échec *)} \\
& \quad \mid \mathsf{c}(z).\ \mathrm{let}\, x = \mathsf{dec}(z, \mathsf{k}(\mathsf{a},\mathsf{b}))\ \mathrm{in}\ \overline{\mathsf{c}}\langle \mathsf{enc}(\mathsf{f}(x), \mathsf{k}(\mathsf{a},\mathsf{b}))\rangle \\
& \};\ \emptyset \\
\rightarrow_{(\mathrm{par})} & \emptyset;\ \{ \\
& \quad \mathrm{new}\, x.\ \overline{\mathsf{c}}\langle\, \mathsf{enc}(x, \mathsf{k}(\mathsf{a},\mathsf{b}))\,\rangle.\ \mathsf{c}(y). \\
& \qquad \mathrm{if}\ \mathsf{dec}(y, \mathsf{k}(\mathsf{a},\mathsf{b})) = \mathsf{f}(x)\ \mathrm{then}\, 0\ \text{(* succès *)} \\
& \qquad \mathrm{else}\, 0\ \text{(* échec *)}\ , \\
& \quad \mathsf{c}(z).\ \mathrm{let}\, x = \mathsf{dec}(z, \mathsf{k}(\mathsf{a},\mathsf{b}))\ \mathrm{in}\ \overline{\mathsf{c}}\langle \mathsf{enc}(\mathsf{f}(x), \mathsf{k}(\mathsf{a},\mathsf{b}))\rangle \\
& \};\ \emptyset \\
\rightarrow_{(\mathrm{new})} & \{n\};\ \{ \\
& \quad \overline{\mathsf{c}}\langle\, \mathsf{enc}(n, \mathsf{k}(\mathsf{a},\mathsf{b}))\,\rangle.\ \mathsf{c}(y). \\
& \qquad \mathrm{if}\ \mathsf{dec}(y, \mathsf{k}(\mathsf{a},\mathsf{b})) = \mathsf{f}(n)\ \mathrm{then}\, 0\ \text{(* succès *)} \\
& \qquad \mathrm{else}\, 0\ \text{(* échec *)}\ , \\
& \quad \mathsf{c}(z).\ \mathrm{let}\, x = \mathsf{dec}(z, \mathsf{k}(\mathsf{a},\mathsf{b}))\ \mathrm{in}\ \overline{\mathsf{c}}\langle \mathsf{enc}(\mathsf{f}(x), \mathsf{k}(\mathsf{a},\mathsf{b}))\rangle \\
& \};\ \emptyset \\
\rightarrow_{(\mathrm{comm})} & \{n\};\ \{ \\
& \quad \mathsf{c}(y). \\
& \qquad \mathrm{if}\ \mathsf{dec}(y, \mathsf{k}(\mathsf{a},\mathsf{b})) = \mathsf{f}(n)\ \mathrm{then}\, 0\ \text{(* succès *)} \\
& \qquad \mathrm{else}\, 0\ \text{(* échec *)}\ , \\
& \quad \mathrm{let}\, x = \mathsf{dec}(\mathsf{enc}(n, \mathsf{k}(\mathsf{a},\mathsf{b})), \mathsf{k}(\mathsf{a},\mathsf{b}))\ \mathrm{in} \\
& \qquad \overline{\mathsf{c}}\langle \mathsf{enc}(\mathsf{f}(x), \mathsf{k}(\mathsf{a},\mathsf{b}))\rangle \\
& \};\ \emptyset \\
\rightarrow_{(\mathrm{comm})} & \{n\};\ \{ \\
& \quad \mathrm{if}\ \mathsf{dec}(\mathsf{enc}(\mathsf{f}(\mathsf{dec}(\mathsf{enc}(n, \mathsf{k}(\mathsf{a},\mathsf{b})), \mathsf{k}(\mathsf{a},\mathsf{b}))), \mathsf{k}(\mathsf{a},\mathsf{b})), \mathsf{k}(\mathsf{a},\mathsf{b})) = \mathsf{f}(n) \\
& \quad \mathrm{then}\, 0\ \text{(* succès *)} \\
& \quad \mathrm{else}\, 0\ \text{(* échec *)}\ , \\
& \quad 0 \\
& \};\ \emptyset \\
\rightarrow_{(\text{test-1})} & \{n\};\ \{\ 0\ \text{(* succès *)}\ ,\ 0\ \};\ \emptyset \\
\rightarrow_{(\mathrm{nil})}\rightarrow_{(\mathrm{nil})} & \{n\};\ \emptyset;\ \emptyset
\end{array}
$$

Tab. 3.2 – Détails des transitions pour une session du protocole Handshake

Paires	$\mathsf{fst}(\mathsf{pair}(x,y)) \doteq x$ $\mathsf{snd}(\mathsf{pair}(x,y)) \doteq y$
Chiffrement symétrique déterministe	$\mathsf{dec}(\mathsf{enc}(x,y),y) \doteq x$
Chiffrement symétrique déterministe surjectif (cipher)	$\mathsf{dec}(\mathsf{enc}(x,y),y) \doteq x$ $\mathsf{enc}(\mathsf{dec}(x,y),y) \doteq x$
Chiffrement symétrique probabiliste avec intégrité	$\mathsf{sdec}(\mathsf{senc}(x,y,z),y) \doteq x$ $\mathsf{sdec\_success}(\mathsf{senc}(x,y,z),y) \doteq \mathsf{ok}$
Chiffrement asymétrique probabiliste avec intégrité	$\mathsf{pdec}(\mathsf{penc}(x,\mathsf{pub}(y),z),y) \doteq x$ $\mathsf{pdec\_success}(\mathsf{penc}(x,\mathsf{pub}(y),z),y) \doteq \mathsf{ok}$
Chiffrement asymétrique probabiliste avec intégrité, mais révélant la clé (publique) de chiffrement	$\mathsf{pdec}(\mathsf{penc}(x,\mathsf{pub}(y),z),y) \doteq x$ $\mathsf{pdec\_success}(\mathsf{penc}(x,\mathsf{pub}(y),z),y) \doteq \mathsf{ok}$ $\mathsf{getkey}(\mathsf{penc}(x,y',z)) \doteq y'$
Chiffrement symétrique probabiliste sans intégrité, révélant les égalités entre clés de chiffrement	$\mathsf{sdec}(\mathsf{senc}(x,y,z),y) \doteq x$ $\mathsf{samekey}(\mathsf{senc}(x,y,z),\ \mathsf{senc}(x',y,z')) \doteq \mathsf{ok}$
Signature probabiliste et contenant le message signé en clair	$\mathsf{check}(\mathsf{sig}(x,y,z),\mathsf{pub}(y)) \doteq \mathsf{ok}$ $\mathsf{getmsg}(\mathsf{sig}(x,y,z)) \doteq x$
Fonction de hachage	symbole libre h d'arité 1
Empreinte idempotente	$\mathsf{h}(\mathsf{h}(x)) \doteq \mathsf{h}(x)$
Involution	$\mathsf{f}(\mathsf{f}(x)) \doteq x$

TAB. 3.3 – Exemples de théories équationnelles sous-terme-convergentes

Chiffrement symétrique déterministe surjectif par bloc (*block-cipher* en mode ECB [BR03])	$\mathsf{dec}(\mathsf{enc}(x,y),y) \doteq x$ $\mathsf{enc}(\mathsf{dec}(x,y),y) \doteq x$ $\mathsf{fst}(\mathsf{pair}(x,y)) \doteq x$ $\mathsf{snd}(\mathsf{pair}(x,y)) \doteq y$ $\mathsf{enc}(\mathsf{pair}(x,y),z) \doteq \mathsf{pair}(\mathsf{enc}(x,z),\mathsf{enc}(y,z))$ $\mathsf{dec}(\mathsf{pair}(x,y),z) \doteq \mathsf{pair}(\mathsf{dec}(x,z),\mathsf{dec}(y,z))$
Signature en aveugle (*blind signature*) [KR05, AC06]	$\mathsf{checksign}(\mathsf{sign}(x,y),\mathsf{pub}(y)) \doteq x$ $\mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x,y),z),y) \doteq \mathsf{sign}(x,z)$
Chiffrement symétrique, déterministe et commutatif, par ex. basé sur RSA [SRA81]	$\mathsf{pdec}(\mathsf{penc}(x,y),y) \doteq x$ $\mathsf{penc}(\mathsf{penc}(x,y),z) \doteq \mathsf{penc}(\mathsf{penc}(x,z),y)$
Empreinte (*checksum*) respectant le OU exclusif, par ex. un code de redondance cyclique (CRC) [CDL06, DLLT06]	$x \oplus y \doteq y \oplus x$ $(x \oplus y) \oplus z \doteq x \oplus (y \oplus z)$ $x \oplus 0 \doteq x$ $x \oplus x \doteq 0$ $\mathsf{h}(x \oplus y) \doteq \mathsf{h}(x) \oplus \mathsf{h}(y)$
Exponentiation modulaire Diffie-Hellman [DH76, AF01]	$x \cdot y \doteq y \cdot x$ $(x \cdot y) \cdot z \doteq x \cdot (y \cdot z)$ $\mathsf{exp}(\mathsf{exp}(a,x),y) \doteq \mathsf{exp}(a, x \cdot y)$

Tab. 3.4 – Exemples de théories équationnelles arbitraires

Les primitives probabilistes sont ainsi modélisées à l'aide d'un argument supplémentaire z, destiné à recevoir un nombre aléatoire, typiquement généré par un new.

Par intégrité du chiffrement, on entend ici la capacité pour le détenteur d'un message chiffré d'observer le succès ou l'échec du déchiffrement pour cause de mauvaise clé. Notons que dans certains cas, cette propriété découle de la théorie équationnelle sans qu'il soit nécessaire d'ajouter une équation spécifique. Par exemple, dans le cas du chiffrement symétrique déterministe (table 3.3, ligne 2), on peut distinguer les deux scénarios par rechiffrement : si $t = \mathsf{enc}(x, y)$ alors $\mathsf{enc}(\mathsf{dec}(t, y), y) = t$ mais $\mathsf{enc}(\mathsf{dec}(t, z), z) \neq t$.

La seconde table regroupe différents exemples plus évolués. De nombreuses autres théories équationnelles sont bien sûr pertinentes en cryptographie. On pourra consulter V. Cortier *et al.* [CDL06] pour un panorama des propriétés algébriques les plus étudiées. Dans le chapitre 5, nous considérons également la théorie des groupes abéliens, tandis que le chapitre 6 détaille plus particulièrement le cas du OU exclusif ainsi que différentes primitives de chiffrement.

En ce qui concerne la structure du langage proposé, l'exemple précédent (3.1) suggère que notre langage dispose de toutes les constructions nécessaires pour modéliser l'exécution d'un protocole cryptographique. Toutefois, plusieurs aspects peuvent être discutés à la lumière des langages et des modèles existants.

Fonctions partielles. De nombreux langages tirés du spi-calcul [Bor01, Bla02, Cor02, BN05, BB05] modélisent des fonctions cryptographiques partielles, c'est-à-dire dont l'exécution peut échouer — par exemple le déchiffrement en présence d'un mécanisme d'intégrité. Par souci de simplicité, nous avons omis une telle construction dans le langage. Il est généralement admis (voir par exemple M. Abadi et C. Fournet [AF01]) que les propriétés de sécurité d'un protocole restent inchangées si l'on modélise chaque fonction partielle par une fonction totale complétée d'un prédicat testant le *succès* de l'application — comme nous l'avons fait dans la table 3.3. Par exemple, plutôt que modéliser le (dé)chiffrement symétrique par une règle de réécriture conditionnelle à la manière de [Bla02, Cor02, BB05] :

$$\mathsf{sdec}(x, y) \quad \rightarrow \quad \begin{cases} x' & \text{si } x \text{ est de la forme } \mathsf{senc}(x', y, z') \\ \bot & \text{sinon} \end{cases}$$

on préférera ici une modélisation sous la forme d'un symbole total plus un prédicat testant le succès du déchiffrement :

$$\begin{aligned} \mathsf{sdec}(\mathsf{senc}(x, y, z), y) &\doteq x \\ \mathsf{sdec\_success}(\mathsf{senc}(x, y, z), y) &\doteq \mathsf{ok} \end{aligned}$$

Outre une certaine uniformisation, l'intérêt de ce formalisme est de permettre des raisonnements équationnels dans lesquels la stratégie d'évaluation des expressions n'est pas spécifiée a priori.

Notons qu'à l'inverse, modéliser une fonction totale par un destructeur partiel n'est pas sûr en général [DJ04a]. En effet, un tel modèle est susceptible d'oublier certains termes : par exemple le déchiffrement d'un message par une clé erronée peut ou non être une valeur correcte, selon les implémentations.

Filtrage des messages. Historiquement, une manière répandue de modéliser les fonctions partielles est d'autoriser certaines formes de filtrage sur les messages. Ainsi, le spi-calcul [AG97] dispose-t-il d'une construction (avec nos notations)

$$\text{case } u \text{ of } \mathsf{senc}(x, v) \text{ in } P$$

permettant de tester si le message u est de la forme $\mathsf{senc}(t, v)$ pour un certain terme t (u et v étant fixés et clos au moment où cette instruction est exécutée). Lorsque c'est le cas, t est le déchiffrement de u par v ; le processus se poursuit alors par $P\{x \mapsto t\}$.

Dans le cas évoqué, recourir au filtrage permet de se passer du symbole de déchiffrement sdec (et de la règle de réécriture associée) au prix d'une construction spécifique dans le langage. La forme contraignante du filtrage entraîne que cette construction n'est ni plus ni moins expressive que la règle de réécriture conditionnelle ci-dessus.

Plus généralement, il est remarquable que les premiers modèles développés pour l'analyse (logique) de protocoles reposent quasi-exclusivement[1] sur des algèbres de termes libres, c'est-à-dire sans théorie équationnelle. C'est le cas du spi-calcul [AG97], mais aussi de l'*approche inductive* de L. Paulson [Pau98], des *strand spaces* [THG99] et des modèles utilisés à l'origine par les outils NRL [Mea96] et FDR [Low96] ou par les outils à base d'automates d'arbre [Mon03, Gou00, CCM01] et de clauses de Horn [Bla01, Gou04, GRV05].

Dans tous les modèles mentionnés ci-dessus excepté le spi-calcul, les capacités de l'intrus sont spécifiées par un ensemble de règles de déduction tandis que les participants honnêtes ont accès (implicitement) au filtrage arbitraire des messages du protocole. Par exemple, si $\{\_\}\_$ représente une opération de chiffrement usuelle (disons symétrique déterministe), le protocole suivant est exprimable et analysable dans ces modèles :

$$\begin{array}{lll} 0. & A \to B : & \{0\}_N \\ 1. & B \to A : & N \end{array}$$

On note qu'ici, le participant honnête B doit extraire la clé fraîche N du premier message pour la renvoyer à A.

Cette exemple précis peut encore être modélisé dans notre formalisme à l'aide d'un symbole *privé* spécial getkey et de l'équation :

$$\mathsf{getkey}(\mathsf{enc}(x, y)) \doteq y$$

(en plus de l'équation usuelle $\mathsf{dec}(\mathsf{enc}(x, y), y) \doteq x$).

Toutefois, en présence de filtrage *et* de théories équationnelles (voir par exemple les premiers travaux sur le OU exclusif [CKRT03b, CLS03], sur la primitive Diffie-Hellman [MS03, CKRT03a] et sur les groupes abéliens avec ou sans homomorphisme [DLLT06]), S. Delaune [Del06b] note que l'ajout de tels symboles privés n'est pas toujours réalisable, car pouvant conduire à une théorie équationnelle dégénérée, c'est-à-dire où tous les termes sont égaux. Dans ce cas, le modèle équationnel avec filtrage apparaît donc strictement plus expressif que le modèle équationnel considéré ici, dit « avec tests d'égalité » [Del06b].

Dans la mesure où l'expressivité additionnelle obtenue ne nous paraît pas strictement nécessaire aux applications, et par souci de simplicité[2], nous n'autorisons

[1] Toutefois, le modèle original de D. Dolev et A. Yao [DY83] est énoncé en termes d'équations.

[2] En présence de théorie équationnelle, les opérations de filtrage doivent être restreintes afin de garantir leur caractère déterministe [CKRT03b, CLS03, DLLT06].

pas le filtrage des messages dans le langage proposé. De ce point de vue, notre modèle est donc comparable au pi-calcul appliqué et aux autres modèles équationnels sans filtrage [DJ04a, CR05].

Dans les chapitres 5 et 6, nous verrons que ce formalisme est également bien adapté à une interprétation calculatoire.

Noms et constantes libres. Le formalisme initial du pi-calcul appliqué [AF01] requiert une théorie équationnelle seulement stable par renommage des noms. Par souci de cohérence avec les autres chapitres, nous préférons ici assimiler les noms à des constantes libres. Autrement dit, par la proposition 2.5, $=_E$ est supposée stable par remplacement quelconque des noms — à la manière de M. Abadi et V. Cortier [AC04, AC06]. Notons qu'il reste possible de modéliser des classes distinctes de valeurs aléatoires dans ce formalisme en utilisant au choix :

– un système de sortes approprié, ou
– des symboles unaires et des prédicats de tests dédiés.

Par exemple, dans le second cas, pour différencier clés publiques, privées et symétriques, on utilise les symboles unaires pubk, prk, symk et $\mathsf{is\_pubk}$, $\mathsf{is\_prk}$, $\mathsf{is\_symk}$, munis des équations :

$$\begin{aligned} \mathsf{is\_pubk}(\mathsf{pubk}(x)) &\doteq \mathsf{ok} \\ \mathsf{is\_prk}(\mathsf{prk}(x)) &\doteq \mathsf{ok} \\ \mathsf{is\_symk}(\mathsf{symk}(x)) &\doteq \mathsf{ok} \end{aligned}$$

où ok est une constante dédiée. Les équations relatives aux chiffrements symétrique et asymétrique deviennent par exemple :

$$\begin{aligned} \mathsf{sdec}(\mathsf{senc}(x, \mathsf{symk}(y), z), \mathsf{symk}(y)) &\doteq x \\ \mathsf{pdec}(\mathsf{penc}(x, \mathsf{pubk}(y), z), \mathsf{prk}(y)) &\doteq x \end{aligned}$$

3.1.4 Équivalences observationnelles usuelles ($\cong_{\mathbf{bf}}$, $\cong_{\mathbf{b}}$, $\cong_{\mathbf{may}}$)

La sémantique opérationnelle précédente décrit le comportement d'une configuration seule. Pour modéliser le fait qu'un attaquant puisse ou non distinguer deux configurations $\mathcal{K}_1$ et $\mathcal{K}_2$ l'une de l'autre, nous définissons différentes relations d'équivalence entre configurations, appelées *équivalences observationnelles*.

Ces relations s'inspirent (plus ou moins fidèlement) des équivalences classiques du spi-calcul et du pi-calcul appliqué [AG97, AF01]. Chacune correspond à un certain pouvoir de discernement de l'intrus. Néanmoins, toutes les relations ont en commun de représenter les actions de l'attaquant par une configuration publique $\mathcal{L}$ arbitraire en interaction avec la configuration $\mathcal{K}_i$ ($i \in \{1, 2\}$). La condition « $\mathcal{L}$ publique » correspond au fait que l'attaquant n'a pas accès aux noms de $\mathcal{K}_i$, ni aux symboles privés. Intuitivement, le résultat de l'interaction de l'attaquant avec la configuration $\mathcal{K}_i$ est l'envoi (ou non) d'un message sur un certain canal public u. Formellement, ceci se traduit par les définitions classiques suivantes.

Soit $\mathcal{K} = \mathcal{E}; \mathcal{P}_\mathrm{e}; \mathcal{P}_\mathrm{a}$ une configuration et u un terme tel que $\mathrm{names}(u) \subseteq \mathcal{E}$. On dit que la configuration $\mathcal{K}$ peut *émettre sur le canal* u si $\mathcal{P}_\mathrm{e}$ contient un processus de la forme $\bar{v}\langle t\rangle.\, P$ avec $u =_E v$. On note cette propriété $\mathcal{K}{\downarrow_u}$. On écrit $\mathcal{K}{\Downarrow_u}$ lorsqu'il existe une configuration $\mathcal{K}'$ telle que $\mathcal{K} \to^* \mathcal{K}'{\downarrow_u}$.

Définition 3.2 (May-équivalence)**.** Deux configurations sont may-équivalentes, ce que l'on note $\mathcal{K}_1 \cong_{\text{may}} \mathcal{K}_2$, lorsque pour toute configuration publique $\mathcal{L}$ et pour tout terme public clos $u \in \mathcal{F}_{\text{pub}}[\emptyset]$,

$$\mathcal{L} \,|\, \mathcal{K}_1 \Downarrow_u \text{ équivaut à } \mathcal{L} \,|\, \mathcal{K}_2 \Downarrow_u .$$

Selon les applications, une autre notion d'équivalence, plus fine et souvent plus commode à prouver, est également intéressante.

Définition 3.3 (Congruence barbue)**.** Une relation $\sim$ entre configurations est une *pré-congruence barbue* si elle est symétrique et si pour tout $\mathcal{K}_1 \sim \mathcal{K}_2$, les points suivants sont satisfaits :

- pour tout terme public clos u, $\mathcal{K}_1 \downarrow_u$ implique $\mathcal{K}_2 \Downarrow_u$;
- si $\mathcal{K}_1 \to \mathcal{K}_1'$ alors il existe $\mathcal{K}_2'$ tel que $\mathcal{K}_2 \to^* \mathcal{K}_2'$ et $\mathcal{K}_1' \sim \mathcal{K}_2'$;
- pour toute configuration publique $\mathcal{L}$, on a $\mathcal{L} \,|\, \mathcal{K}_1 \sim \mathcal{L} \,|\, \mathcal{K}_2$.

La *congruence barbue*, notée $\cong_{\text{b}}$, est la plus grande pré-congruence barbue. On montre que $\cong_{\text{b}}$ est bien réflexive et transitive. (Ces faits sont justifiés par le lemme 3.1 ci-dessous.)

On définit les *pré-congruences barbues fortes* et la *congruence barbue forte*, notée $\cong_{\text{bf}}$, de manière identique en remplaçant ci-dessus les occurrences de $\Downarrow_u$ par $\downarrow_u$, et celles de $\to^*$ par $\to$.

Les pré-congruences barbues vérifient notamment les propriétés classiques suivantes :

Lemme 3.1.

(i) La relation vide et l'égalité sont des pré-congruences barbues.

(ii) L'union d'une famille (quelconque) de pré-congruences barbues est une pré-congruence barbue.

(iii) Si $\sim_1$ et $\sim_2$ sont deux pré-congruences barbues alors $(\sim_1 \sim_2) \cup (\sim_2 \sim_1)$ est une pré-congruence barbue.

(iv) Les clôtures réflexive et transitive d'une pré-congruence barbue sont des pré-congruences barbues.

De plus, toutes les propriétés ci-dessus restent vraies pour les pré-congruences barbues fortes.

Les notions de may-équivalence et de congruence barbue introduites ici sont — intuitivement — proches des notions correspondantes du pi-calcul appliqué [AF01]. Notons toutefois qu'en l'absence d'équivalence structurelle, notre congruence barbue forte est très restreinte : par exemple $0|0 \not\cong_{\text{bf}} 0$ (comparer le nombre de transitions).

3.1.5 Bi-processus et équivalence associée $\cong_{\text{diff}}$

Nous décrivons maintenant une dernière équivalence observationnelle inspirée des travaux de B. Blanchet, M. Abadi et C. Fournet [BAF05], et de F. Pottier et V. Simonet [PS03, Pot02]. Il s'agit d'une relation basée sur les bi-processus, c'est-à-dire une paire de processus ne différant que par les valeurs qu'il contiennent. Nous montrons plus bas (théorème 3.4) que cette relation est une certaine pré-congruence

barbue forte, intuitivement guidée par la structure des processus. Pour cette raison, cette notion d'équivalence s'avère particulièrement bien adaptée à l'analyse automatique [BAF05]. Plus loin (section 3.4), nous étudions la décidabilité de cette notion pour des processus *finis*, c'est-à-dire sans réplication.

Un *processus double* (ou *bi-processus*) P^2 est un processus dans lequel chaque terme u (ou v) apparaissant dans la syntaxe est dédoublé en un couple de termes u_1 et u_2 de même sorte, noté $\mathrm{diff}[u_1, u_2]$. Une telle expression $u^2 = \mathrm{diff}[u_1, u_2]$ est appelée *terme double*. Formellement :

P^2, Q^2	$::=$		processus double
	$\mid$	0	processus vide
	$\mid$	$P^2 \mid Q^2$	composition parallèle
	$\mid$	$!\, P^2$	réplication
	$\mid$	$u^2(x).\, P^2$	réception sur le canal u
	$\mid$	$\overline{u^2}\langle v^2\rangle.\, P^2$	émission de v sur le canal u
	$\mid$	if $u^2 = v^2$ then P^2 else Q^2	test d'égalité
	$\mid$	new $x.\, P^2$	création de noms
	$\mid$	wait $.P^2$	attente jusqu'à la phase suivante

Les *projections* d'un terme double u^2, notées $\mathsf{fst}(u^2)$ et $\mathsf{snd}(u^2)$, sont définies de manière attendue par

$$\begin{aligned} \mathsf{fst}(\mathrm{diff}[u_1, u_2]) &= u_1 \\ \mathsf{snd}(\mathrm{diff}[u_1, u_2]) &= u_2 \end{aligned}$$

De même, les projections d'un bi-processus P^2, notées $\mathsf{fst}(P^2)$ et $\mathsf{snd}(P^2)$, sont obtenues en remplaçant toutes les expressions $\mathrm{diff}[u_1, u_2]$ respectivement par u_1 et u_2 dans P^2.

On étend les relations d'égalité et de différence modulo E aux termes doubles (de même sorte) de la manière suivante :

$$\begin{aligned} u^2 =^2_E v^2 \quad &\text{ssi} \quad \mathsf{fst}(u^2) =_E \mathsf{fst}(v^2) \text{ et } \mathsf{snd}(u^2) =_E \mathsf{snd}(v^2) \\ u^2 \neq^2_E v^2 \quad &\text{ssi} \quad \mathsf{fst}(u^2) \neq_E \mathsf{fst}(v^2) \textit{ et } \mathsf{snd}(u^2) \neq_E \mathsf{snd}(v^2) \end{aligned}$$

Notons que certains termes doubles u^2 et v^2 ne sont donc reliés ni par $=^2_E$ ni par $\neq^2_E$. On note $u^2 \uparrow_E v^2$ lorsque c'est le cas, c'est-à-dire quand

$$\mathsf{fst}(u^2) =_E \mathsf{fst}(v^2) \iff \mathsf{snd}(u^2) =_E \mathsf{snd}(v^2).$$

Convention. Par la suite, afin d'alléger les notations, on omettra les indices ^{2} lorsque cela ne prête pas à confusion. On identifie chaque terme (simple) u avec le terme double $\mathrm{diff}[u, u]$. De même, les processus (simples) sont vus comme des processus doubles particuliers.

Les termes doubles sont munis d'une structure de $\mathcal{F}$-algèbre de manière naturelle :

$$f(\mathrm{diff}[t_1, t'_1], \ldots, \mathrm{diff}[t_n, t'_n]) \triangleq \mathrm{diff}[f(t_1, \ldots, t_n), f(t'_1, \ldots, t'_n)]$$

On bénéficie également de la notion de substitution double, *i.e.* une fonction σ des variables dans les termes doubles. L'application d'une substitution double est définie par $\mathsf{fst}(t\sigma) = \mathsf{fst}(t)\,\mathsf{fst}(\sigma)$ et $\mathsf{snd}(t\sigma) = \mathsf{snd}(t)\,\mathsf{snd}(\sigma)$. Ainsi, on a par exemple

$$\mathrm{diff}[t_1, t_2]\{x \mapsto \mathrm{diff}[u_1, u_2]\} = \mathrm{diff}[t_1\{x \mapsto u_1\}, t_2\{x \mapsto u_2\}]$$

Un *nom double* est un terme double n tel que $\mathsf{fst}(n)$ et $\mathsf{snd}(n)$ sont des noms.

$$\mathcal{E};\ \{0\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{(nil)}$$

$$\mathcal{E};\ \{P \,|\, Q\,\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \{P,\, Q\,\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{(par)}$$

$$\mathcal{E};\ \{!\,P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \{P,\, !\,P\,\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{(repl)}$$

$$\mathcal{E};\ \{u(x).\,P,\ \overline{v}\langle t\rangle.\,Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \{P\{x \mapsto t\},\, Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{(comm)}$$
$$\text{si } \begin{cases} \mathrm{sort}(x) = \mathrm{sort}(t) \\ u =^2_E v \end{cases}$$

$$\mathcal{E};\ \{\text{if } u = v \text{ then } P \text{ else } Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \{P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{si } u =^2_E v \qquad \text{(test-1)}$$

$$\mathcal{E};\ \{\text{if } u = v \text{ then } P \text{ else } Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \{Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{si } u \neq^2_E v \qquad \text{(test-2)}$$

$$\mathcal{E};\ \{\text{new } x.P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \{n\} \cup \mathcal{E};\ \{P\{x \mapsto n\}\,\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \qquad \text{(new)}$$
$$\text{si } \begin{cases} \mathrm{sort}(n) = \mathrm{sort}(x) \\ n \in \mathcal{N} - \mathsf{fst}(\mathcal{E}) - \mathsf{snd}(\mathcal{E}) \end{cases}$$

$$\mathcal{E};\ \{\text{wait}\,.P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \{P\} \uplus \mathcal{P}_{\mathrm{a}} \qquad \text{(wait)}$$

$$\mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \to \mathcal{E};\ \mathcal{P}_{\mathrm{a}};\ \emptyset \qquad \text{si } \mathcal{P}_{\mathrm{a}} \neq \emptyset \qquad \text{(next)}$$

Tab. 3.5 – Transitions entre configurations doubles

$$\mathcal{E};\ \{u(x).\,P,\ \overline{v}\langle t\rangle.\,Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \quad \uparrow \qquad \text{si } \begin{cases} \mathrm{sort}(x) = \mathrm{sort}(t) \\ u \uparrow_E v \end{cases} \qquad \text{(div-comm)}$$

$$\mathcal{E};\ \{\text{if } u = v \text{ then } P \text{ else } Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \quad \uparrow \qquad \text{si } u \uparrow_E v \qquad \text{(div-test)}$$

Tab. 3.6 – Configurations doubles divergentes

Définition 3.4. Une *pré-configuration double* est un triplet $\mathcal{K} = \mathcal{E}; \mathcal{P}_{\mathrm{e}}; \mathcal{P}_{\mathrm{a}}$ où
- $\mathcal{E}$ est un ensemble fini de noms doubles ;
- $\mathcal{P}_{\mathrm{e}}$ et $\mathcal{P}_{\mathrm{a}}$ sont des multi-ensembles finis de bi-processus clos.

Les projections $\mathsf{fst}(\mathcal{K})$ et $\mathsf{snd}(\mathcal{K})$ sont définies de la manière attendue.

Un tel triplet $\mathcal{K} = \mathcal{E}; \mathcal{P}_{\mathrm{e}}; \mathcal{P}_{\mathrm{a}}$ est une *configuration double* ssi $\mathrm{names}(\mathsf{fst}(\mathcal{P}_{\mathrm{e}}; \mathcal{P}_{\mathrm{a}})) \subseteq \mathsf{fst}(\mathcal{E})$, $\mathrm{names}(\mathsf{snd}(\mathcal{P}_{\mathrm{e}}; \mathcal{P}_{\mathrm{a}})) \subseteq \mathsf{snd}(\mathcal{E})$ et si l'ensemble $\{(n_1, n_2) \mid \mathrm{diff}[n_1, n_2] \in \mathcal{E}\}$ est une injection partielle.

L'opérateur | est étendu aux configurations doubles comme précédemment : si $\mathcal{K} = \mathcal{E}; \mathcal{P}_{\mathrm{e}}; \mathcal{P}_{\mathrm{a}}$ et $\mathcal{K} = \mathcal{E}'; \mathcal{P}'_{\mathrm{e}}; \mathcal{P}'_{\mathrm{a}}$, alors $\mathcal{K} \,|\, \mathcal{K}' \triangleq \mathcal{E} \uplus \mathcal{E}';\ \mathcal{P}_{\mathrm{e}} \uplus \mathcal{P}'_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}} \uplus \mathcal{P}'_{\mathrm{a}}$.

Nous étendons la relation de transition à petits pas aux configurations doubles grâce aux règles de la table 3.5. L'extension est conservative dans le sens où pour toute configuration simple $\mathcal{K}$, on a $\mathcal{K} \to \mathcal{K}'$ selon la table 3.5 ssi $\mathcal{K}'$ est simple et $\mathcal{K} \to \mathcal{K}'$ d'après la table 3.1. Une différence visible avec les transitions simples est l'utilisation des relations $=^2_E$ et $\neq^2_E$. Du fait que ces deux relations ne sont pas complémentaires, le système de transition possède des configurations doubles

$\mathcal{K}$ dites *divergentes* (table 3.6), ce que l'on note $\mathcal{K}\uparrow$. La relation $\rightarrow^*\uparrow$ est notée $\Uparrow$.

À l'aide de ces notions, nous pouvons définir une nouvelle équivalence observationnelle entre configurations.

Définition 3.5 (diff-équivalence)**.** Une configuration double $\mathcal{K}$ est *sans divergence* ssi pour toute configuration publique (simple) $\mathcal{L}$, $\mathcal{L} \mid \mathcal{K} \not\Uparrow$.

Deux configurations (simples) $\mathcal{K}_1$ et $\mathcal{K}_2$ sont *diff-équivalentes*, ce que l'on note $\mathcal{K}_1 \cong_{\text{diff}} \mathcal{K}_2$, ss'il existe une configuration double sans divergence $\mathcal{K}$ telle que $\mathsf{fst}(\mathcal{K}) = \mathcal{K}_1$ et $\mathsf{snd}(\mathcal{K}) = \mathcal{K}_2$.

Notons que $\cong_{\text{diff}}$ est bien réflexive, symétrique et transitive (notamment parce que c'est le cas pour la relation de non-divergence $\not\Uparrow$ vue comme une relation binaire entre configurations simples de même structure : $\mathsf{fst}(\mathcal{K}) \approx^0 \mathsf{snd}(\mathcal{K})$ ssi $\mathcal{K} \not\Uparrow$).

Montrons également que la diff-équivalence est stable par renommage (bijectif) des noms d'une configuration.

Définition 3.6. On note $\cong_{\text{n}}$ la relation d'*équivalence par renommage* entre configurations, définie par : $\mathcal{K}_1 \cong_{\text{n}} \mathcal{K}_2$ ss'il existe un renommage ρ allant de $\text{names}(\mathcal{K}_1)$ vers $\text{names}(\mathcal{K}_2)$ tel que $\mathcal{K}_2 = \mathcal{K}_1\rho$.

Proposition 3.2. *La diff-équivalence est stable par renommage :* $\cong_n \subseteq \cong_{diff}$.

Démonstration. Soit $\mathcal{K}$ une configuration double telle que $\mathsf{fst}(\mathcal{K}) \cong_{\text{n}} \mathsf{snd}(\mathcal{K})$. Par analyse des règles, il est immédiat de vérifier que $\mathcal{K} \rightarrow \mathcal{K}'$ implique $\mathsf{fst}(\mathcal{K}') \cong_{\text{n}} \mathsf{snd}(\mathcal{K}')$. De plus, si $\mathcal{L}$ est une configuration publique, alors $\mathsf{fst}(\mathcal{L} \mid \mathcal{K}) \cong_{\text{n}} \mathsf{snd}(\mathcal{L} \mid \mathcal{K})$. Enfin, comme $=_E$ est stable par renommage des noms (et donc également $\neq_E$), on a $\mathcal{K} \not\Uparrow$. □

Convention. Dans la suite, grâce à la proposition précédente, nous nous se restreignons sans perte de généralité à l'étude des configurations doubles $\mathcal{K} = \mathcal{E}; \mathcal{P}_{\text{e}}; \mathcal{P}_{\text{a}}$ telles que $\mathcal{E}$ est un ensemble de noms simples.

Par conséquent, dans les transitions entre configurations doubles, la règle de création de noms se simplifie en :

$$\mathcal{E};\, \{\text{new}\, x.P\} \uplus \mathcal{P}_{\text{e}};\, \mathcal{P}_{\text{a}} \rightarrow \{n\} \cup \mathcal{E};\, \{P\{x \mapsto n\}\} \uplus \mathcal{P}_{\text{e}};\, \mathcal{P}_{\text{a}} \qquad \text{(new')}$$
$$\text{si} \begin{cases} \text{sort}(n) = \text{sort}(x) \\ n \in \mathcal{N} - \mathcal{E} \end{cases}$$

Nous étendons la relation de renommage aux configurations doubles (de la forme simplifiée ci-dessus) de manière naturelle : $\mathcal{K}_1 \cong_{\text{n}} \mathcal{K}_2$ ss'il existe un renommage $\rho : \text{names}(\mathcal{K}_1) \rightarrow \text{names}(\mathcal{K}_2)$ tel que $\mathcal{K}_2 = \mathcal{K}_1\rho$. Le lemme simple suivant s'établit par une preuve similaire à celle de la proposition 3.2.

Lemme 3.3. *Les relations* $\rightarrow$ *et* $\uparrow$ *sur les configurations doubles sont compatibles avec le renommage, au sens où* $\mathcal{K}_1 \cong_n \mathcal{K}'_1$ *implique*

$$\mathcal{K}_1\uparrow \Leftrightarrow\ \mathcal{K}'_1\uparrow$$
$$\forall \mathcal{K}_2, \quad \mathcal{K}_1 \rightarrow \mathcal{K}_2\ \Rightarrow\ \exists \mathcal{K}'_2,\ \mathcal{K}'_1 \rightarrow \mathcal{K}'_2\ \textit{et}\ \mathcal{K}_2 \cong_n \mathcal{K}'_2$$

3.1.6 Relations entre équivalences observationnelles

Nous concluons cette section par l'étude des relations entre les équivalences observationnelles définies jusqu'à présent.

Théorème 3.4. *On a les inclusions suivantes :*

$$\cong_{diff} \subseteq \cong_{bf} \subseteq \cong_{b} \subseteq \cong_{may} .$$

Démonstration. Les inclusions $\cong_{\mathrm{bf}} \subseteq \cong_{\mathrm{b}} \subseteq \cong_{\mathrm{may}}$ découlent directement des définitions. Pour montrer $\cong_{\mathrm{diff}} \subseteq \cong_{\mathrm{bf}}$, on vérifie que $\cong_{\mathrm{diff}}$ est une pré-congruence barbue forte.

Supposons que $\mathcal{K}_1 \cong_{\mathrm{diff}} \mathcal{K}_2$, c'est-à-dire qu'il existe une configuration double $\mathcal{K}$ tel que $\mathsf{fst}(\mathcal{K}) = \mathcal{K}_1$, $\mathsf{snd}(\mathcal{K}) = \mathcal{K}_2$ et que $\mathcal{K}$ est sans divergence, au sens où pour toute configuration simple publique $\mathcal{L}$, $\mathcal{L}|\mathcal{K} \not\Uparrow$. Par la proposition 3.2, on peut supposer que $\mathrm{names}(\mathcal{K}_1) = \mathrm{names}(\mathcal{K}_2)$.

- Par symétrie de la sémantique double, on a également $\mathcal{K}_2 \cong_{\mathrm{diff}} \mathcal{K}_1$.
- Si u est un terme public clos tel que $\mathcal{K}_1 \downarrow_u$, alors étant donné $\mathcal{L} = \emptyset; u(x).0; \emptyset$, $\mathcal{L}|\mathcal{K} \not\downarrow$ implique $\mathcal{K}_2 \downarrow_u$.
- Si $\mathcal{K}_1 \to \mathcal{K}_1'$, par inspection des règles, $\mathcal{K} \not\downarrow$ implique qu'il existe $\mathcal{K}'$ et $\mathcal{K}_2' = \mathsf{snd}(\mathcal{K}')$ tels que $\mathcal{K} \to \mathcal{K}'$, $\mathsf{fst}(\mathcal{K}') = \mathcal{K}_1'$ et $\mathcal{K}_2 \to \mathcal{K}_2'$. Comme $\mathcal{K}$ est sans divergence, $\mathcal{K}'$ l'est également ; autrement dit $\mathcal{K}_1' \cong_{\mathrm{diff}} \mathcal{K}_2'$.
- Pour toute configuration simple publique $\mathcal{L}$, $\mathcal{K}_1 \cong_{\mathrm{diff}} \mathcal{K}_2$ implique bien $\mathcal{L}|\mathcal{K}_1 \cong_{\mathrm{diff}} \mathcal{L}|\mathcal{K}_2$. En effet, dans le cas contraire, sachant que $\mathsf{fst}(\mathcal{L}|\mathcal{K}) = \mathcal{L}|\mathcal{K}_1$ et $\mathsf{snd}(\mathcal{L}|\mathcal{K}) = \mathcal{L}|\mathcal{K}_2$, il existe $\mathcal{L}'$ telle que $\mathcal{L}'|(\mathcal{L}|\mathcal{K}) = (\mathcal{L}'|\mathcal{L})|\mathcal{K} \Uparrow$. □

Comme on peut l'imaginer, les inclusions précédentes sont strictes. Comparé aux autres équivalences, $\cong_{\mathrm{diff}}$ correspond intuitivement à un intrus capable d'observer la structure interne des configurations ; seules les valeurs des expressions contenues dans ces configurations lui sont inaccessibles. Par exemple, en règle générale, $P|Q \not\cong_{\mathrm{diff}} Q|P$ tandis cette équivalence est toujours vraie pour $\cong_{\mathrm{bf}}$.

La congruence barbue forte $\cong_{\mathrm{bf}}$ se distingue des équivalences plus générales $\cong_{\mathrm{b}}$ et $\cong_{\mathrm{may}}$, dans la mesure où le nombre de transitions dans une séquence est observable. Par exemple, $!\,0 \not\cong_{\mathrm{bf}} 0$ tandis que $!\,0 \cong_{\mathrm{b}} 0$ (comparer les transitions).

Enfin, la congruence barbue $\cong_{\mathrm{b}}$, à la différence de $\cong_{\mathrm{may}}$, n'est pas une équivalence basée sur les traces, et dépend localement des choix possibles à partir d'une configuration. Ainsi, étant donné trois constantes publiques a, b, c, et l'abréviation $P + Q \triangleq \mathrm{new}\, x.(\overline{x}\langle\rangle \mid x().P \mid x().Q)$ permettant de coder un choix non-déterministe, on dispose du contre-exemple classique suivant :

$$\begin{array}{lccc} & \mathsf{a}().(\mathsf{b}() + \mathsf{c}()) & \cong_{\mathrm{may}} & \mathsf{a}().\mathsf{b}() + \mathsf{a}().\mathsf{c}() \\ \text{mais} & \mathsf{a}().(\mathsf{b}() + \mathsf{c}()) & \not\cong_{\mathrm{b}} & \mathsf{a}().\mathsf{b}() + \mathsf{a}().\mathsf{c}() \end{array}$$

3.2 Propriétés de sécurité

Nous montrons maintenant comment spécifier certaines propriétés de sécurité logique d'un protocole cryptographique au moyen du langage de processus et des outils introduits dans la section précédente.

Au préalable, il nous faut introduire la notion habituelle de contexte de processus. De manière informelle, un *contexte de processus* (ou simplement *contexte*

dans cette section) est un processus dont un des sous-processus est remplacé par un trou, noté _. Formellement, les contextes de processus sont définis de la manière suivante :

C, D	::=		
	$\mid$	$\_$	trou
	$\mid$	$C \mid Q$	composition parallèle (1)
	$\mid$	$P \mid C$	composition parallèle (2)
	$\mid$	$!\,C$	réplication
	$\mid$	$u(x).C$	réception sur le canal u
	$\mid$	$\overline{u}\langle v\rangle.C$	émission de v sur le canal u
	$\mid$	if $u = v$ then C else Q	test d'égalité (1)
	$\mid$	if $u = v$ then P else C	test d'égalité (2)
	$\mid$	new $x.C$	création de noms
	$\mid$	wait $.C$	attente

On note $C[P]$ le processus obtenu en remplaçant _ par P dans un contexte C. L'ensemble des variables définies par un contexte au niveau du trou _ est calculé inductivement par :

$$\begin{aligned}
\mathrm{scope}(_) &= \emptyset \\
\mathrm{scope}(u(x).C) &= \mathrm{scope}(\mathrm{new}\, x.C) \\
&= \mathrm{scope}(C) \cup \{x\} \\
\mathrm{scope}(\overline{u}\langle v\rangle.C) &= \mathrm{scope}(C|Q) \\
&= \mathrm{scope}(P|C) \\
&= \mathrm{scope}(!\,C) \\
&= \mathrm{scope}(\text{if } u = v \text{ then } C \text{ else } Q) \\
&= \mathrm{scope}(\text{if } u = v \text{ then } P \text{ else } C) \\
&= \mathrm{scope}(\text{wait } .C) \\
&= \mathrm{scope}(C)
\end{aligned}$$

Le lemme suivant (que l'on prouve facilement par induction sur C) permet de relier simplement les fonctions var et scope.

Lemme 3.5. *Si $P = C[Q]$, alors*

$$\mathrm{var}(P) = \mathrm{var}(C) \cup (\mathrm{var}(Q) - \mathrm{scope}(C)).$$

Notamment $\mathrm{var}(P) = \emptyset$ *équivaut à* $\mathrm{var}(Q) \subseteq \mathrm{scope}(C)$ *et* $\mathrm{var}(C) = \emptyset$.

Plus généralement, on définit les contextes n-aires de manière similaire, comme des processus dans lesquels exactement n sous-processus sont remplacés par des trous notés $\__1$, $\__2$, …, $\__n$. On note $C[P_1, \ldots, P_n]$ le processus obtenu en remplaçant chaque $\__i$ par P_i dans C. L'ensemble des variables définies au niveau de chaque trou $\__i$ est calculé par une fonction scope_i définie de la même manière que ci-dessus.

Nous énonçons maintenant différentes propriétés de sécurité sur un protocole que l'on suppose modélisé au préalable par un certain processus P.

Exemple 3.2. Considérons à nouveau le protocole Handshake (voir l'exemple 3.1) :

$$\begin{array}{lll} 0. & A \to B : & \{N\}_{k_{AB}} \\ 1. & B \to A : & \{f(N)\}_{k_{AB}} \end{array}$$

Nous avons modélisé ce protocole à l'aide des processus suivants :

$$\begin{array}{rcl} P_A(a,b) & \triangleq & \mathsf{new}\, x.\ \overline{\mathsf{c}}\langle\, \mathsf{enc}(x, \mathsf{k}(a,b))\,\rangle.\ \mathsf{c}(y). \\ & & \quad \mathsf{if}\, \mathsf{dec}(y, \mathsf{k}(a,b)) = \mathsf{f}(x)\ \mathsf{then}\, 0\ \text{(* succès *)} \\ & & \quad \mathsf{else}\, 0\ \text{(* échec *)} \\ P_B(a,b) & \triangleq & \mathsf{c}(z).\ \mathsf{let}\, x = \mathsf{dec}(z, \mathsf{k}(a,b))\ \mathsf{in}\ \overline{\mathsf{c}}\langle \mathsf{enc}(\mathsf{f}(x), \mathsf{k}(a,b))\rangle \\ P_0 & \triangleq & !\,\mathsf{c}(a).\ \mathsf{c}(b).\ P_A(a,b) \quad | \quad !\,\mathsf{c}(a).\ \mathsf{c}(b).\ P_B(a,b) \\ & & |\, !\,\mathsf{c}(a).\ \overline{\mathsf{c}}\langle k(a,\mathsf{i})\rangle.\ \overline{\mathsf{c}}\langle k(\mathsf{i},a)\rangle \end{array}$$

Dans la suite, nous utiliserons également une variante dans laquelle les sessions de l'intrus sont différenciées des autres :

$$\begin{array}{rcl} P & \triangleq & !\,\mathsf{c}(a).\ \mathsf{c}(b).\ \mathsf{if}\, a \neq \mathsf{i}\ \mathsf{and}\ b \neq \mathsf{i}\ \mathsf{then}\, P_A(a,b) \\ & & |\, !\,\mathsf{c}(a).\ \mathsf{c}(b).\ \mathsf{if}\, a \neq \mathsf{i}\ \mathsf{and}\ b \neq \mathsf{i}\ \mathsf{then}\, P_B(a,b) \\ & & |\, !\,\mathsf{c}(a).\ P_A(a,\mathsf{i}) \quad | \quad !\,\mathsf{c}(b).\ P_B(\mathsf{i},b) \\ & & |\, !\,\mathsf{c}(a).\ \overline{\mathsf{c}}\langle k(a,\mathsf{i})\rangle.\ \overline{\mathsf{c}}\langle k(\mathsf{i},a)\rangle \end{array}$$

La troisième ligne permet aux agents honnêtes de communiquer avec l'intrus malgré tout. Notons que dans cette exemple-ci, la partie $!\,\mathsf{c}(b).\ P_B(\mathsf{i},b)$ est redondante avec les capacités de l'intrus.

3.2.1 Secret simple

La propriété la plus naturelle à formaliser est sans doute le secret simple, c'est-à-dire le fait qu'aucun attaquant ne puisse deviner en totalité la valeur d'une certaine expression [DY83].

Définition 3.7. Soit $P = C[Q]$ un processus clos, bad et c deux constantes libres, respectivement privée et publique, et n'apparaissant pas dans P. Pour que la définition ait un sens, on suppose que la sorte de bad n'est pas dégénérée dans E. Soit t un terme tel que $\mathrm{var}(t) \subseteq \mathrm{scope}(C)$.

Le *secret simple* de t est vérifié à partir du point Q dans $P = C[Q]$ si pour toute configuration publique $\mathcal{L}$,

$$\mathcal{L}|P^* \not\Downarrow_{\mathsf{bad}}$$

où $P^* \triangleq C[\,Q \,|\, \mathsf{c}(x).\ \mathsf{if}\, x = t\ \mathsf{then}\, \overline{\mathsf{bad}}\langle\rangle\,]$.

Exemple 3.3 (suite de 3.2)**.** Dans le cas du protocole Handshake, on modélise le secret simple de toutes les clés partagées, au choix, en modifiant P en deux endroits

comme dans la définition ci-dessus :

$$\begin{aligned} P^* \triangleq\ & !\,\mathsf{c}(a).\ \mathsf{c}(b).\ \text{if } a \neq \mathsf{i} \text{ and } b \neq \mathsf{i} \text{ then} \\ & \quad (P_A(a,b) \mid \mathsf{c}(x).\ \text{if } x = \mathsf{k}(a,b) \text{ then } \overline{\mathsf{bad}}\langle\rangle) \\ & \mid !\,\mathsf{c}(a).\ \mathsf{c}(b).\ \text{if } a \neq \mathsf{i} \text{ and } b \neq \mathsf{i} \text{ then} \\ & \quad (P_B(a,b) \mid \mathsf{c}(x).\ \text{if } x = \mathsf{k}(a,b) \text{ then } \overline{\mathsf{bad}}\langle\rangle) \\ & \mid !\,\mathsf{c}(a).\ P_A(a,\mathsf{i}) \quad \mid \quad !\,\mathsf{c}(b).\ P_B(\mathsf{i},b) \\ & \mid !\,\mathsf{c}(a).\ \overline{\mathsf{c}}\langle k(a,\mathsf{i})\rangle.\ \overline{\mathsf{c}}\langle k(\mathsf{i},a)\rangle \end{aligned}$$

ou alors plus simplement — dans le cas présent — en modifiant P_0 directement :

$$\begin{aligned} P_0^* \triangleq\ & !\,\mathsf{c}(a).\ \mathsf{c}(b).\ P_A(a,b) \mid !\,\mathsf{c}(a).\ \mathsf{c}(b).\ P_B(a,b) \\ & \mid !\,\mathsf{c}(a).\ \overline{\mathsf{c}}\langle k(a,\mathsf{i})\rangle.\ \overline{\mathsf{c}}\langle k(\mathsf{i},a)\rangle \\ & \mid !\,\mathsf{c}(a).\ \mathsf{c}(b).\ \mathsf{c}(x).\ \text{if } a \neq \mathsf{i} \text{ and } b \neq \mathsf{i} \text{ and } x = \mathsf{k}(a,b) \text{ then } \overline{\mathsf{bad}}\langle\rangle \end{aligned}$$

On peut montrer (par exemple à l'aide de l'outil Proverif [Blab]) que le secret simple des clés partagées entre agents honnêtes est effectivement vérifié pour ce protocole.

3.2.2 Authentification

Nous nous plaçons maintenant dans le cas d'un protocole où deux agents A et B souhaitent s'authentifier mutuellement et éventuellement s'accorder sur des données. Nous proposons une modélisation dans notre langage des notions d'authentification injective et non injective dues à G. Lowe [Low97], faisant suite aux travaux de Woo et Lam sur les *assertions de correspondance* [WL93].

Soit begin, end, bad des constantes libres, privées et de sorte non dégénérée.

Nous supposons fixé un processus $T_{\text{non-inj}}$ écoutant sur les canaux begin et end, tel que $T_{\text{non-inj}}$ envoie le signal bad ss'il existe un message t reçu sur le canal end qui n'a pas été précédé par au moins un message $t' =_E t$ sur le canal begin.

De même, soit T_{inj} un processus écoutant sur les canaux begin et end, tel que T_{inj} envoie le signal bad ssi à un moment donné, il existe un terme t tel que le nombre de messages $t' =_E t$ reçus sur le canal end par T_{inj} est strictement supérieur au nombre de messages $t' =_E t$ reçus sur le canal begin. (Nous détaillons plus bas de tels processus $T_{\text{non-inj}}$ et T_{inj}.)

Définition 3.8. Soit P un processus clos modélisant le protocole tel que begin, end et bad n'apparaissent pas dans P. Pour simplifier, on suppose que P est en une phase (*i.e.* sans wait). On suppose de plus que

- $P = C[Q, R]$ où Q et R sont les points d'entrée respectifs des deux rôles A et B dans le protocole, et que
- $R = C'[0]$ où le 0 correspond au point de sortie du rôle B en cas de succès (apparent) du protocole.[3]

Soit t un terme vérifiant $\text{var}(t) \subseteq \text{scope}_1(C) \cap \text{scope}(C[Q, C'])$ que A souhaite authentifier du point de vue de B. Par exemple, t contient les noms ou les clés publiques de A et de B, ainsi que des clés de session, un horadatage (*timestamp*)... etc.

On dit que le terme t est *authentifié non injectivement* (respectivement *authentifié injectivement*) par A auprès de B ssi pour toute configuration publique

[3]Le cas où B aurait plusieurs points de sortie en cas de succès se traite de manière similaire.

$\mathcal{L}$,

$$\mathcal{L} \mid P^*_{\text{non-inj}} \not\Vdash_{\mathsf{bad}}$$

(respectivement $\mathcal{L} \mid P^*_{\text{inj}} \not\Vdash_{\mathsf{bad}}$), où les processus $P^*_{\text{non-inj}}$ et P^*_{inj} sont définis comme suit :

$$\begin{aligned} P^*_{\text{non-inj}} &\triangleq C\left[\,\overline{\mathsf{begin}}\langle t\rangle.\, Q,\; C'[\overline{\mathsf{end}}\langle t\rangle]\,\right] \mid T_{\text{non-inj}} \\ P^*_{\text{inj}} &\triangleq C\left[\,\overline{\mathsf{begin}}\langle t\rangle.\, Q,\; C'[\overline{\mathsf{end}}\langle t\rangle]\,\right] \mid T_{\text{inj}} \end{aligned}$$

Cette définition correspond aux notions d'authentification injective ou non injective (*agreement*) de A vis-à-vis de B proposées par G. Lowe [Low97] dans le cas où le terme t contient les noms de A et de B et les données de session. Si t ne contient que les noms de A et de B, on parle d'authentification faible (*weak agreement*). Si t ne contient que le nom de A, on parle seulement de vivacité (*aliveness*).

Exemple 3.4 (suite de 3.2)**.** Dans le cas du protocole Handshake, la propriété d'authentification (respectivement injective ou non injective) de B par rapport à A s'exprime de la manière suivante :

$$\begin{aligned} P^*_{\text{inj/non-inj}} \triangleq\ & !\,\mathsf{c}(a).\ \mathsf{c}(b).\ \text{if } a \neq \mathsf{i} \text{ and } b \neq \mathsf{i} \text{ then } P^*_A(a,b) \\ \mid\ & !\,\mathsf{c}(a).\ \mathsf{c}(b).\ \text{if } a \neq \mathsf{i} \text{ and } b \neq \mathsf{i} \text{ then } P^*_B(a,b) \\ \mid\ & !\,\mathsf{c}(a).\ P_A(a,\mathsf{i}) \quad \mid \quad !\,\mathsf{c}(b).\ P_B(\mathsf{i},b) \\ \mid\ & !\,\mathsf{c}(a).\ \overline{\mathsf{c}}\langle k(a,\mathsf{i})\rangle.\ \overline{\mathsf{c}}\langle k(\mathsf{i},a)\rangle \\ \mid\ & T_{\text{inj/non-inj}} \end{aligned}$$

où les rôles de agents ont été instrumentés en suivant la définition 3.8 pour donner :

$$\begin{aligned} P^*_A(a,b) \triangleq\ & \text{new } x.\ \overline{\mathsf{c}}\langle\, \mathsf{enc}(x, \mathsf{k}(a,b))\,\rangle.\ \mathsf{c}(y). \\ & \quad \text{if } \mathsf{dec}(y, \mathsf{k}(a,b)) = \mathsf{f}(x) \text{ then } \overline{\mathsf{end}}\langle \mathsf{h}(a,b,\mathsf{f}(x))\rangle \text{ (* succès *)} \\ & \quad \text{else } 0 \text{ (* échec *)} \\ P^*_B(a,b) \triangleq\ & \mathsf{c}(z).\ \text{let } x = \mathsf{dec}(z, \mathsf{k}(a,b)) \text{ in } \overline{\mathsf{begin}}\langle \mathsf{h}(a,b,\mathsf{f}(x))\rangle.\ \overline{\mathsf{c}}\langle \mathsf{enc}(\mathsf{f}(x), \mathsf{k}(a,b))\rangle \end{aligned}$$

Notons que les processus $P_A(a,i)$ et $P_B(i,b)$ ne sont pas instrumentés. En effet, les canaux begin et end ne sont pas accessibles à l'intrus (et rien ne l'obligerait à les utiliser à bon escient).

Dans le cas présent, on peut montrer que les propriétés d'authentification (non)-injectives sont effectivement vérifiées.

Codage de l'authentification. Nous donnons maintenant une description possible des processus $T_{\text{non-inj}}$ et T_{inj}. Ce codage permet de modéliser l'authentification tout en gardant le même langage par la suite. À des fins de vérification pratique, une extension simple du langage (comme par exemple [Bla02]) serait naturellement préférable.

On suppose que la signature $\mathcal{F}$ et la théorie équationnelle E comportent (ou sont augmentés par) des symboles permettant de représenter des listes de couples de termes, par exemple, les symboles nil, $\mathsf{cons}(3)$, $\mathsf{head}_1(1)$, $\mathsf{head}_2(1)$, $\mathsf{tail}(1)$ munis

des équations suivantes :[4]

$$\begin{aligned} \mathsf{head}_1(\mathsf{cons}(x,y,z)) &\doteq x \\ \mathsf{head}_2(\mathsf{cons}(x,y,z)) &\doteq y \\ \mathsf{tail}(\mathsf{cons}(x,y,z)) &\doteq z \end{aligned}$$

On définit alors $T_{\text{non-inj}}$ et T_{inj} comme suit.

$$\begin{aligned} T_1(\mathit{events}) &\triangleq \mathit{events}(z).\ \mathsf{begin}(y).\ \overline{\mathit{events}}\langle\mathsf{cons}(\mathsf{begin},y,z)\rangle \\ T_2(\mathit{events}) &\triangleq \mathit{events}(z).\ \mathsf{end}(y).\ \overline{\mathit{events}}\langle\mathsf{cons}(\mathsf{end},y,z)\rangle \end{aligned}$$

$$\begin{aligned} S_1(\mathit{events}) &\triangleq \mathit{events}(z).\ \text{if } \mathsf{head}_1(z)=\mathsf{end} \text{ then new } \mathit{nomatch} \text{ in} \\ &\quad (\ !\,S_2(\mathit{nomatch},\mathsf{head}_2(z)) \mid \overline{\mathit{nomatch}}\langle\mathsf{tail}(z)\rangle\ ) \\ S_2(\mathit{nomatch},y) &\triangleq \mathit{nomatch}(z).\ \text{if } z=\mathsf{nil} \text{ then } \overline{\mathsf{bad}}\langle\rangle \text{ else} \\ &\quad \text{if } \mathsf{head}_1(z)=\mathsf{begin} \text{ and } \mathsf{head}_2(z)=y \text{ then } 0 \text{ else} \\ &\quad \overline{\mathit{nomatch}}\langle\mathit{tail}(z)\rangle \end{aligned}$$

$$\begin{aligned} R_1(\mathit{events}) &\triangleq \mathit{events}(z).\ \text{if } \mathsf{head}_1(z)=\mathsf{end} \text{ then new } \mathit{cnt1},\mathit{cnt2} \text{ in} \\ &\quad (\ !\,R_2(\mathit{cnt1},\mathit{cnt2},\mathsf{head}_2(z)) \\ &\quad \mid \overline{\mathit{cnt1}}\langle\mathsf{cons}(\mathit{nil},\mathit{nil},\mathit{nil})\rangle.\ \overline{\mathit{cnt2}}\langle\mathit{tail}(z)\rangle\ ) \\ R_2(\mathit{cnt1},\mathit{cnt2},y) &\triangleq \mathit{cnt1}(\mathit{ctr}).\ \mathit{cnt2}(z).\ \text{if } z=\mathsf{nil} \text{ then}(\\ &\quad \text{if } \mathit{ctr}\neq\mathsf{nil} \text{ then } \overline{\mathsf{bad}}\langle\rangle \\ &\quad)\,\text{else if } \mathsf{head}_1(z)=\mathsf{end} \text{ and } \mathsf{head}_2(z)=y \text{ then} \\ &\quad \overline{\mathit{cnt1}}\langle\mathsf{cons}(\mathit{nil},\mathit{nil},\mathit{ctr})\rangle.\ \overline{\mathit{cnt2}}\langle\mathit{tail}(z)\rangle \\ &\quad \text{else if } \mathsf{head}_1(z)=\mathsf{begin} \text{ and } \mathsf{head}_2(z)=y; \mathit{ctr}\neq\mathsf{nil} \text{ then} \\ &\quad \overline{\mathit{cnt1}}\langle\mathsf{tail}(\mathit{ctr})\rangle.\ \overline{\mathit{cnt2}}\langle\mathit{tail}(z)\rangle \\ &\quad \text{else if } \mathsf{head}_1(z)=\mathsf{begin} \text{ and } \mathsf{head}_2(z)=y; \mathit{ctr}=\mathsf{nil} \text{ then} \\ &\quad \overline{\mathit{cnt1}}\langle\mathsf{nil}\rangle.\ \overline{\mathit{cnt2}}\langle\mathit{tail}(z)\rangle \end{aligned}$$

et finalement :

$$\begin{aligned} T_{\text{non-inj}} &\triangleq \text{new } \mathit{events}.\ (\ \overline{\mathit{events}}\langle\mathsf{nil}\rangle \mid !\,T_1(\mathit{events}) \mid !\,T_2(\mathit{events}) \mid S_1(\mathit{events})\ ) \\ T_{\text{inj}} &\triangleq \text{new } \mathit{events}.\ (\ \overline{\mathit{events}}\langle\mathsf{nil}\rangle \mid !\,T_1(\mathit{events}) \mid !\,T_2(\mathit{events}) \mid R_1(\mathit{events})\ ) \end{aligned}$$

En deux mots, le rôle de T_1 et T_2 est de construire la liste des événements entrants. S_1 et R_1 parcourent cette liste en partant du message le plus récent. Une fois un événement end trouvé, ces processus appellent (éventuellement) la routine de recherche S_2 sur les événements antérieurs ou la routine de comptage R_2, selon le cas.

[4]Implicitement, on suppose que ces symboles ne vérifient pas d'autres équations, c'est-à-dire qu'il existe une présentation de E dans laquelle seules les équations données utilisent les symboles nil, cons, head_1, head_2 et tail.

Dans le cas du comptage (T_{inj}), les entiers *ctr* sont codés en unaire sous forme de listes : $0 \approx \mathsf{nil}$, $n+1 \approx \mathsf{cons}(\mathsf{nil}, \mathsf{nil}, n)$. Notez que l'on ignore les événements begin lorsque le compteur *ctr* est déjà à 0. Ceci revient à avancer la fin du test avant l'arrivée des begin en excédent.

3.2.3 Attaques par dictionnaire

La plupart des systèmes cryptographiques nécessitent des clés et plus généralement des secrets de longueur suffisante pour être hors de portée d'une recherche exhaustive par l'attaquant. Toutefois, dans certain cas, des considérations pratiques conduisent à relâcher cette hypothèse. On parle alors de *secret faible* ou de *secret à faible entropie*. Un exemple très répandu de secrets faibles est celui des mots de passe choisis par un utilisateur humain.

Les secrets faibles sont potentiellement vulnérables aux *attaques par dictionnaire* (*dictionary attacks, guessing attacks*) consistant à retrouver un secret en essayant successivement toutes les valeurs possibles. (L'ensemble de ces valeurs est parfois appelé *dictionnaire*.) Fort heureusement, tous les secrets faibles ne sont pas attaquables de la sorte. En effet, lors de la recherche exhaustive l'attaquant doit encore pouvoir distinguer la vraie valeur du secret parmi les autres valeurs — typiquement en exploitant certaines redondances entre les messages connus [Low04].

Parmi les attaques par dictionnaire, les attaques *hors-ligne* (*off-line*) se distinguent par le fait que leur phase d'énumération ne nécessite pas d'interaction avec le protocole. Notons que l'attaquant a encore la possibilité d'interagir avec le protocole avant de commencer la recherche exhaustive.

Les attaques par dictionnaire hors-ligne sont plus pertinentes d'un point de vue pratique que le cas général (*on-line*) dans la mesure où, pour des raisons d'efficacité ou de politique de sécurité, un attaquant a rarement la possibilité d'envoyer sur le réseau un nombre de requêtes de l'ordre de grandeur de la taille du dictionnaire. (Lorsque c'est le cas, la plupart des systèmes à base de mots de passe deviennent d'ailleurs immédiatement attaquables.) Dans la suite, nous nous concentrons sur la notion d'attaques par dictionnaire hors-ligne.

Nous proposons maintenant une modélisation des attaques par dictionnaire reposant sur l'instruction wait du langage, dans la lignée de R. Corin *et al.* [CDE04] (pour le cas d'un attaquant passif) et de B. Blanchet *et al.* [BAF05] (pour le cas général).

Définition 3.9. Soit $\cong$ une équivalence observationnelle parmi $\cong_{\text{may}}$, $\cong_{\text{b}}$, $\cong_{\text{bf}}$, $\cong_{\text{diff}}$. Soit $P = C[Q]$ un processus clos en une phase, et c une constante libre publique n'apparaissant pas dans P. Soit t un terme tel que $\text{var}(t) \subseteq \text{scope}(C)$.

Le terme t est *caché hors-ligne* à partir du point Q dans $P = C[Q]$ par rapport à $\cong$ si

$$P_1^* \cong P_2^*$$

où $P_1^* = C[\, Q \mid \text{wait} \,.\, \text{new}\, x.\overline{\mathsf{c}}\langle t \rangle\,]$ avec $x \notin \text{var}(t)$, et $P_2^* = C[\, Q \mid \text{wait} \,.\, \text{new}\, x.\overline{\mathsf{c}}\langle x \rangle\,]$.

Intuitivement, cette propriété signifie qu'une fois la phase active du protocole terminée, l'intrus ne peut pas faire de différence entre le terme secret t et un nombre aléatoire frais x modélisant un élément quelconque du dictionnaire. Autrement dit, l'intrus n'a pas la possibilité d'identifier la vraie valeur du secret t pendant la phase de recherche hors-ligne.

L'instruction additionnelle $\mathsf{new}\, x$ dans P_1^* est nécessaire dans le cas des équivalences $\cong_{\mathrm{bf}}$ et $\cong_{\mathrm{diff}}$ où intuitivement, le nombre d'étapes de réduction voire la structure du processus est observable par l'intrus. Dans le cas où P comporte des instructions wait, la définition s'adapte naturellement en ajoutant suffisamment de wait avant les instructions $\overline{\mathsf{c}}\langle x\rangle$ et $\overline{\mathsf{c}}\langle t\rangle$.

Étant donné la nature hors-ligne des attaques par dictionnaire, on peut s'étonner que la définition 3.9 dépende apparemment de la relation d'équivalence choisie. La proposition suivante montre qu'il n'en est rien.

Proposition 3.6. *Soit $P = C[Q]$ un processus clos en une phase (i.e. sans* wait*) et t un terme tel que* $\mathrm{var}(t) \subseteq \mathrm{scope}(C)$*. Soit $x \notin \mathrm{var}(t)$ et*

$$\begin{aligned} P_1^* &\triangleq C[\,Q \mid \mathsf{wait}\,.\,\mathsf{new}\, x.\,\overline{\mathsf{c}}\langle t\rangle\,] \\ P_2^* &\triangleq C[\,Q \mid \mathsf{wait}\,.\,\mathsf{new}\, x.\,\overline{\mathsf{c}}\langle x\rangle\,] \end{aligned}$$

On a :

$$\begin{aligned} P_1^* \cong_{may} P_2^* &\Leftrightarrow P_1^* \cong_{b} P_2^* \\ &\Leftrightarrow P_1^* \cong_{bf} P_2^* \\ &\Leftrightarrow P_1^* \cong_{diff} P_2^* \end{aligned}$$

Schéma de preuve. Étant donné le théorème 3.4, il suffit de montrer que $P_1^* \cong_{\mathrm{may}} P_2^*$ implique $P_1^* \cong_{\mathrm{diff}} P_2^*$. Par contradiction, supposons que $P_1^* \ncong_{\mathrm{diff}} P_2^*$. Soit

$$P^* \triangleq C[\,Q \mid \mathsf{wait}\,.\,\mathsf{new}\, x.\,\overline{\mathsf{c}}\langle \mathrm{diff}[t, x]\rangle].$$

Nous admettons pour le moment le fait suivant, qui sera prouvé indépendamment dans la section 3.3 (lemme 3.11).

> $P_1^* \ncong_{\mathrm{diff}} P_2^*$ implique qu'il existe un processus simple public L vérifiant $L|P^* \Uparrow$ et de la forme
>
> $$L = C_L[\mathsf{wait}\,.\mathsf{c}(x_1), \ldots, \mathsf{c}(x_n).\,\mathsf{if}\, u = v\ \mathsf{then}\, 0\, \mathsf{else}\, 0]$$
>
> où u et v sont des termes simples, C_L est un contexte construit uniquement à partir des instructions d'envoi et de réception et $n \geq 1$ est dû à la présence éventuelle d'une réplication dans C.

Étant donné la sémantique du wait et la forme de P^* et de L, $L|P^* \Uparrow$ implique l'existence d'une configuration $\mathcal{K} = \mathcal{E};\, \mathcal{P}_{\mathrm{a}}, \emptyset$ telle que $L|P^* \rightarrow^* \mathcal{K}$ et $\mathcal{P}_{\mathrm{a}}$ est de la forme suivante :

$$\mathcal{P}_{\mathrm{a}} = \{\mathsf{c}(x_1) \ldots \mathsf{c}(x_n).\,\mathsf{if}\, u\sigma = v\sigma\ \mathsf{then}\, 0\, \mathsf{else}\, 0,\ \underbrace{\mathsf{new}\, x.\overline{\mathsf{c}}\langle \mathrm{diff}[t\mu, x]\rangle}_{n \text{ fois}}\ \}$$

où μ et σ sont des substitutions partielles, $x \notin \mathrm{var}(t\mu)$, pour tout $1 \leq i \leq n$, $x_i \notin \mathrm{dom}(\sigma) \cup \mathrm{var}(\mathrm{im}(\sigma))$, $u\sigma \neq_E v\sigma$ et $u\sigma\{x_i \mapsto t\mu\}_{1\leq i\leq n} =_E v\sigma\{x_i \mapsto t\mu\}_{1\leq i\leq n}$.

Soit d une constante publique libre fraîche, et $y_1, \ldots, y_n$ des variables fraîches. Considérons le processus

$$L' \triangleq C_L[\mathsf{wait}\,.\,\mathsf{c}(x_1) \ldots \mathsf{c}(x_n).\,\mathsf{new}\, y_1 \ldots \mathsf{new}\, y_n.\ \mathsf{if}\, u = v\ \text{ and }\ u' \neq v'\ \mathsf{then}\, \overline{\mathsf{d}}\langle\rangle\, \mathsf{else}\, 0]$$

où $(u', v') \triangleq (u, v)\{x_1 \mapsto y_1, \ldots, x_n \mapsto y_n\}$.

Sachant que $L|P_1^* = \mathsf{fst}(L|P^*) \rightarrow^* \mathsf{fst}(\mathcal{K})$ et étant donné la forme de $\mathcal{K}$, on a $L'|P_1^* \Downarrow_{\mathsf{d}}$.

Par contradiction, supposons que $L'|P_2^* \Downarrow_{\mathsf{d}}$. On en déduit alors l'existence d'une configuration $\mathcal{K}' = \mathcal{E}'; \mathcal{P}'_{\mathrm{a}}, \emptyset$ telle que $L|P^* \rightarrow^* \mathcal{K}'$, et $\mathcal{P}'_{\mathrm{a}}$ est de la forme

$$\mathcal{P}'_{\mathrm{a}} = \{\mathsf{c}(x_1) \ldots \mathsf{c}(x_n).\, \mathrm{new}\, y_1 \ldots \mathrm{new}\, y_n.\\ \mathrm{if}\, u\sigma' = v\sigma' \ \mathrm{and}\ u'\sigma' \neq v'\sigma' \ \mathrm{then}\, \overline{\mathsf{d}}\langle\rangle\, \mathrm{else}\, 0,\\ \underbrace{\mathrm{new}\, x.\overline{\mathsf{c}}\langle x\rangle}_{n \text{ fois}}\ \}$$

où $n \geq 1$, $x \notin \mathrm{var}(t\mu')$, pour tout $1 \leq i \leq n$, $x_i, y_i \notin \mathrm{dom}(\sigma') \cup \mathrm{var}(\mathrm{im}(\sigma'))$, $u\sigma' =_E v\sigma'$ et $u'\sigma' \neq_E v'\sigma'$. Or, par définition, $u'\sigma' = u\{x_i \mapsto y_i\}_{1\leq i\leq n}\sigma' = u\sigma'\{x_i \mapsto y_i\}_{1\leq i\leq n}$ et de même pour $v'\sigma'$, contradiction. □

Exemple 3.5 (suite de 3.2)**.** Considérons à nouveau le protocole Handshake. Conformément à la définition 3.9, nous modélisons l'existence éventuelle d'attaques par dictionnaire sur les clés partagées par la (non-)équivalence des processus suivants :

$$\begin{aligned} P_1^* \ \triangleq\ & !\,\mathsf{c}(a).\, \mathsf{c}(b).\, P_A(a,b) \mid !\,\mathsf{c}(a).\, \mathsf{c}(b).\, P_B(a,b)\\ & \mid !\,\mathsf{c}(a).\, \overline{\mathsf{c}}\langle k(a,\mathsf{i})\rangle.\, \overline{\mathsf{c}}\langle k(\mathsf{i},a)\rangle\\ & \mid !\,\mathsf{c}(a).\, \mathsf{c}(b).\ \mathrm{if}\, a \neq \mathsf{i}\ \mathrm{and}\ b \neq \mathsf{i}\ \mathrm{then}\, \mathrm{wait}\,.\, \mathrm{new}\, x.\, \overline{\mathsf{c}}\langle \mathsf{k}(a,b)\rangle \end{aligned}$$

$$\begin{aligned} P_2^* \ \triangleq\ & !\,\mathsf{c}(a).\, \mathsf{c}(b).\, P_A(a,b) \mid !\,\mathsf{c}(a).\, \mathsf{c}(b).\, P_B(a,b)\\ & \mid !\,\mathsf{c}(a).\, \overline{\mathsf{c}}\langle k(a,\mathsf{i})\rangle.\, \overline{\mathsf{c}}\langle k(\mathsf{i},a)\rangle\\ & \mid !\,\mathsf{c}(a).\, \mathsf{c}(b).\ \mathrm{if}\, a \neq \mathsf{i}\ \mathrm{and}\ b \neq \mathsf{i}\ \mathrm{then}\, \mathrm{wait}\,.\, \mathrm{new}\, x.\, \overline{\mathsf{c}}\langle x\rangle \end{aligned}$$

Notons que les processus ci-dessus permettent de modéliser une attaque par dictionnaire sur un nombre arbitraire de clés partagées à fois.

En l'occurrence, on peut montrer que $P_1^* \not\approx_{\mathrm{diff}} P_2^*$. Ceci correspond à l'attaque par dictionnaire suivante bien connue [GLNS93, DJ04b] : après une session normale du protocole entre A et B

$$\begin{array}{lll} 0. & A \rightarrow B : & \{N\}_{k_{AB}} \\ 1. & B \rightarrow A : & \{f(N)\}_{k_{AB}} \end{array}$$

l'attaquant connaît les messages : $m_1 = \{N\}_{k_{AB}}$ et $m_2 = \{f(N)\}_{k_{AB}}$. Pour distinguer la bonne valeur de $x = k_{AB}$ des autres valeurs possibles, il lui suffit de tester si $f(dec(m_1, x)) = dec(m_2, x)$, où dec est la fonction de déchiffrement.

3.2.4 Secret fort

La notion précédente de résistance aux attaques par dictionnaire modélise l'impossibilité pour l'attaquant d'acquérir une information exploitable (hors-ligne) sur un secret du protocole. Cette notion est bien adaptée au secret des clés et des mots de passe. Toutefois dans le cas des messages secrets transmis entre agents honnêtes, on peut souhaiter une propriété intuitivement[5] plus forte, appelée secret fort.

[5] Voir la remarque plus loin.

Nous proposons ici une définition inspirée des définitions habituelles [AG97, AF01, Bla04, BAF05], mais exprimable en terme d'équivalence de processus clos.

Définition 3.10. Soit $\cong$ une équivalence observationnelle parmi $\cong_{\text{may}}$, $\cong_{\text{b}}$, $\cong_{\text{bf}}$, $\cong_{\text{diff}}$. Soit $P = C[\text{let}\, x = t\, \text{in}\, Q]$ un processus clos, et c une constante libre publique n'apparaissant pas dans P.

Le secret fort de t dans $P = C[\text{let}\, x = t\, \text{in}\, Q]$ est vérifié par rapport à $\cong$ si

$$P_1^* \cong P_2^*$$

où

$$\begin{array}{rcl} P_1^* & \triangleq & C[\mathsf{c}(x).\, Q\,] \\ P_2^* & \triangleq & C[\mathsf{c}(x').\, Q\{x \mapsto t\}\,] \end{array}$$

et la variable $x' \notin \text{var}(t)$ est de même sorte que x.

Ainsi, une expression est fortement secrète (à certaines occurrences) lorsque l'intrus ne peut distinguer deux exécutions du protocole dans lesquelles cette expression est ou bien laissée telle quelle ou bien remplacée (aux occurrences concernées) par un message choisi par l'intrus.

Remarque. Dans le cas où t est un nom n'apparaissant pas dans Q, et où C est le contexte vide, le secret fort de t dans $P = (\text{let}\, x = t\, \text{in}\, Q)$ implique bien son secret simple (à partir du point P) : en effet, supposons qu'il existe $\mathcal{L}$ tel que

$$\mathcal{L} \mid P \mid (\mathsf{c}'(x).\, \text{if}\, x = t\, \text{then}\, \overline{\mathsf{bad}}\langle\rangle) \Downarrow_{\mathsf{bad}}$$

Alors, en posant

$$\mathcal{L}' = \mathcal{L} \mid (\mathsf{c}'(x).\, \text{if}\, x = t\, \text{then}\, \overline{\mathsf{a}}\langle\rangle) \mid \overline{\mathsf{c}}\langle\mathsf{b}\rangle$$

où a et b sont deux constantes publiques fraîches, on a

$$\mathcal{L}' \mid \mathsf{c}(x').\, Q\{x \mapsto t\} \Downarrow_{\mathsf{a}}$$

mais $\mathcal{L}' \mid \mathsf{c}(x).\, Q \not\Downarrow_{\mathsf{a}}$ car t et a n'apparaissent pas dans $Q\{x \mapsto \mathsf{b}\}$. Par conséquent $\mathsf{c}(x).\, Q \not\cong_{\text{may}} \mathsf{c}(x').\, Q\{x \mapsto t\}$.

Toutefois, on remarque que cette implication n'est pas vraie en général, car par exemple $C[0]$ peut divulguer t tandis que $x \notin \text{var}(Q)$. De cette manière, t n'est pas simplement secret dans $P = C[\text{let}\, x = t\, \text{in}\, Q]$ bien que $P_1^* = P_2^*$.

Plus généralement (et ceci s'applique aussi à la définition originale du secret fort [AG97, Bla04]), il n'est pas aisé de comparer secrets simple et fort car ces notions ne s'appliquent pas rigoureusement aux mêmes objets.

3.2.5 Divulgation des secrets de longue durée

Lorsque les notions de secret précédentes s'appliquent à des données de session, on peut se demander si ces données résistent à la divulgation d'un secret de longue durée du protocole. Par exemple, supposons que un protocole établisse une clé de chiffrement fraîche k_s à l'aide d'une infrastructure de chiffrement à clés publiques $pk_1, \ldots, pk_n$ où $pk_i = pub(sk_i)$. On dira que k_s satisfait une propriété de *secret par anticipation* (*forward secrecy*) [DvOW92, ZD06] si k_s reste secret lorsque les clés privées correspondantes $sk_1, \ldots, sk_n$ sont divulguées après coup — le mot

secret étant pris dans un des sens précédents (secret simple, absence d'attaques par dictionnaire ou secret fort).

Dans notre langage, la divulgation retardée d'un secret s se modélise naturellement à l'aide du sous-processus wait $.\overline{\mathsf{c}}\langle s\rangle$, de manière identique à la définition 3.9. Les notions de secret précédentes se déclinent alors en autant de variantes possibles.

3.2.6 Codage des propriétés de sûreté dans $\cong_{\mathbf{diff}}$

Le lemme suivant nous permet de voir les problèmes de sûreté précédents sur les processus simples comme un cas particulier de la non-divergence de processus doubles.

Lemme 3.7. *Soit $\mathcal{K}$ une configuration simple telle que la constante libre privée* bad *apparaisse dans $\mathcal{K}$ uniquement dans le sous-processus* $\overline{\mathsf{bad}}\langle\rangle$. *Soit* c_0 *et* c_1 *deux constantes publiques de même sorte que* bad *et telles que* $\mathsf{c}_0 \neq_E \mathsf{c}_1$. *Soit* $\mathcal{K}' = \mathcal{K}\{\mathsf{bad} \mapsto \mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]\}$ *la configuration double obtenue en remplaçant* bad *par* $\mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]$ *dans $\mathcal{K}$. Les deux conditions suivantes sont équivalentes :*

(i) Il existe une configuration publique simple $\mathcal{L}$ telle que $\mathcal{L} \,|\, \mathcal{K} \Downarrow_{\mathsf{bad}}$.

(ii) Il existe une configuration publique simple $\mathcal{L}'$ telle que $\mathcal{L}' \,|\, \mathcal{K}' \Uparrow$.

Démonstration. Soit $\mathcal{K}_1$ une configuration simple quelconque contenant la constante libre privée bad uniquement dans le sous-processus $\overline{\mathsf{bad}}\langle\rangle$ (*).

Par inspection des règles, si $\mathcal{K}_1 \to \mathcal{K}_2$, alors $\mathcal{K}_2$ vérifie la condition (*) et $\mathcal{K}_1\{\mathsf{bad} \mapsto \mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]\} \to \mathcal{K}_2\{\mathsf{bad} \mapsto \mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]\}$. De plus, si $\mathcal{K}_1 \downarrow_{\mathsf{bad}}$, par définition l'un de processus en cours d'exécution dans $\mathcal{K}_1$ est $\overline{\mathsf{bad}}\langle\rangle$. On a donc $\mathcal{K}_1\{\mathsf{bad} \mapsto \mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]\} \,|\, \mathsf{c}_0(x).0 \uparrow$ par la règle (div-comm) appliquée à $\overline{\mathsf{bad}}\langle\rangle$ et $\mathsf{c}_0(x)$ (où x est une variable quelconque de même sorte que bad). On en déduit (i) $\Rightarrow$ (ii) par induction sur la longueur de la dérivation $\mathcal{L} \,|\, \mathcal{K} \to^* \downarrow_{\mathsf{bad}}$, en prenant $\mathcal{L}' = \mathcal{L} \,|\, \mathsf{c}_0(x).0$ et initialement $\mathcal{K}_1 = \mathcal{L} \,|\, \mathcal{K}$.

Réciproquement, étant donné une configuration simple quelconque $\mathcal{K}_1$ vérifiant (*), posons $\mathcal{K}_1' = \mathcal{K}_1\{\mathsf{bad} \mapsto \mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]\}$. Si $\mathcal{K}_1' \to \mathcal{K}_2'$ et $\mathcal{K}_1' \not\downarrow$, alors par analyse des règles il existe $\mathcal{K}_2$ vérifiant la condition (*), tel que $\mathcal{K}_1 \to \mathcal{K}_2$ et $\mathcal{K}_2' = \mathcal{K}_2\{\mathsf{bad} \mapsto \mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]\}$. De plus, si $\mathcal{K}_1' \uparrow$, alors étant donné la forme de $\mathcal{K}_1$, la règle (div-comm) est applicable sur le bi-processus $\overline{\mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]}\langle\rangle$ dans $\mathcal{K}_1'$, donc $\mathcal{K}_1 \downarrow_{\mathsf{bad}}$. On en déduit (ii) $\Rightarrow$ (i) par induction sur la longueur de la dérivation $\mathcal{L}' \,|\, \mathcal{K}' \to^* \uparrow$, en prenant $\mathcal{L} = \mathcal{L}'$ et initialement $\mathcal{K}_1 = \mathcal{L} \,|\, \mathcal{K}$. □

Dans la suite de ce chapitre, nous nous concentrons donc sur le problème de la divergence d'une configuration double. Cette notion permet d'énoncer toutes les propriétés de sécurité définies ci-dessus — éventuellement de manière approchée dans le cas du secret fort pour une notion d'équivalence plus générale que $\cong_{\mathrm{diff}}$.

3.3 Sémantique ouverte

Une difficulté importante dans l'étude des propriétés de sécurité définies plus haut est la présence de quantifications universelles sur les configurations publiques en parallèle avec le processus étudié. Nous définissons maintenant une nouvelle sémantique dite *ouverte* $\to_o$ permettant de raisonner directement sur une configuration, simple ou double, en interaction avec un environnement arbitraire.

$$\mathcal{E};\, \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a};\, \mathcal{S} \rightarrow_o \mathcal{E}';\, \mathcal{P}'_\mathrm{e};\, \mathcal{P}'_\mathrm{a};\, \mathcal{S} \qquad (\rightarrow)$$
$$\text{si } \mathcal{E};\, \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a} \rightarrow \mathcal{E}';\, \mathcal{P}'_\mathrm{e};\, \mathcal{P}'_\mathrm{a}$$

$$\mathcal{E};\, \{u(x).\, P\} \uplus \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a};\, \mathcal{S} \rightarrow_o \mathcal{E};\, \{P\{x \mapsto t\}\} \uplus \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a};\, \mathcal{S} \qquad (\text{recv})$$
$$\text{si } \begin{cases} t \in \mathcal{F}_\mathrm{pub}[\mathcal{S}] \\ \exists v \in \mathcal{F}_\mathrm{pub}[\mathcal{S}],\ u =^2_E v \end{cases}$$

$$\mathcal{E};\, \{\overline{u}\langle t\rangle.\, Q\} \uplus \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a};\, \mathcal{S} \rightarrow_o \mathcal{E};\, \{Q\} \uplus \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a};\, \{t\} \cup \mathcal{S} \qquad (\text{send})$$
$$\text{si } \exists v \in \mathcal{F}_\mathrm{pub}[\mathcal{S}],\ u =^2_E v$$

Tab. 3.7 – Transitions entre configurations doubles étendues

$$\mathcal{E};\, \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a};\, \mathcal{S} \ \uparrow_o \quad \text{si } \mathcal{E};\, \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a} \uparrow \qquad (\text{div-comm/-test})$$
$$\mathcal{E};\, \{u(x).\, P\} \uplus \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a};\, \mathcal{S} \ \uparrow_o \quad \text{si } \exists v \in \mathcal{F}_\mathrm{pub}[\mathcal{S}],\ u \uparrow_E v \qquad (\text{div-recv})$$
$$\mathcal{E};\, \{\overline{u}\langle t\rangle.\, P\} \uplus \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a};\, \mathcal{S} \ \uparrow_o \quad \text{si } \exists v \in \mathcal{F}_\mathrm{pub}[\mathcal{S}],\ u \uparrow_E v \qquad (\text{div-send})$$
$$\mathcal{E};\, \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a};\, \mathcal{S} \ \uparrow_o \quad \text{si } \exists u, v \in \mathcal{F}_\mathrm{pub}[\mathcal{S}],\ u \uparrow_E v \qquad (\text{div-static})$$

Tab. 3.8 – Configurations doubles étendues divergentes

3.3.1 Définition de la sémantique ouverte

Nous commençons par augmenter chaque configuration double par un ensemble de termes doubles clos $\mathcal{S}$, représentant intuitivement les messages connus par l'attaquant, *i.e.* potentiellement utilisables par une configuration publique en parallèle.

Définition 3.11. On appelle *configuration double étendue* un quadruplet $\mathcal{I} = \mathcal{E};\, \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a};\, \mathcal{S} = \mathcal{K};\, \mathcal{S}$ où
- $\mathcal{K} = \mathcal{E};\, \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a}$ est une configuration double,
- $\mathcal{S}$ est un ensemble de termes doubles clos,

tels que $\mathrm{names}(\mathcal{P}_\mathrm{e}, \mathcal{P}_\mathrm{a}, \mathcal{S}) \subseteq \mathcal{E}$. Rappelons que nous supposons désormais que $\mathcal{E}$ est un ensemble (fini) de noms simples.

Une configuration double « normale » $\mathcal{K} = \mathcal{E};\, \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a}$ est vue comme une configuration étendue

$$\mathcal{I}(\mathcal{K}) \triangleq \mathcal{E};\, \mathcal{P}_\mathrm{e};\, \mathcal{P}_\mathrm{a};\, \emptyset.$$

Étant donné un ensemble de termes doubles $\mathcal{S}$, rappelons que la notation $\mathcal{F}_\mathrm{pub}[\mathcal{S}]$ désigne l'ensemble des termes doubles que l'on peut obtenir par application de symboles (simples) de $\mathcal{F}_\mathrm{pub}$ à partir des éléments de $\mathcal{S}$.

La sémantique ouverte de notre langage consiste en une relation de transition $\rightarrow_o$ sur les configurations étendues décrite par la table 3.7, ainsi qu'en un prédicat de divergence $\uparrow_o$ défini par la table 3.8. On note $\Uparrow_o$ le prédicat $\rightarrow_o^* \uparrow_o$.

La première règle signifie que $\rightarrow_o$ est une extension de la relation $\rightarrow$. La règle additionnelle (recv) permet à l'attaquant d'envoyer un message quelconque t sur le canal u à destination d'un processus en cours d'exécution, pourvu que u et t

soient déductibles à partir de l'ensemble des termes connus $\mathcal{S}$. La seconde règle additionnelle (send) permet à l'attaquant de recevoir un message t sur le canal u lorsque u est connu ; t est alors ajouté à $\mathcal{S}$.

Concernant les configurations divergentes, deux nouvelles règles (div-recv) et (div-send) rendent compte des cas de divergence dus aux communications entre la configuration testée et l'attaquant. Enfin, la règle (div-static) correspond aux calculs et aux tests que l'attaquant peut effectuer hors-ligne, c'est-à-dire sans interaction avec la configuration testée.

Remarque (Lien avec l'équivalence statique). Supposons que l'on étiquette les éléments d'un ensemble de termes doubles $\mathcal{S}$ par des variables, autrement dit, que l'on se donne une substitution injective double σ d'image $\mathcal{S}$. Lorsque $\mathcal{S} = \mathrm{im}(\sigma)$ ne diverge pas par la règle (div-static), c'est-à-dire quand

$$\text{pour tout } u, v \in \mathcal{F}_{\mathrm{pub}}[\mathcal{S}], \quad \mathsf{fst}(u) =_E \mathsf{fst}(v) \Leftrightarrow \mathsf{snd}(u) =_E \mathsf{snd}(v),$$

on dit que les substitutions $\mathsf{fst}(\sigma)$ et $\mathsf{snd}(\sigma)$ sont *statiquement équivalentes* [AF01]. Étant donné $\sigma_1 = \mathsf{fst}(\sigma)$ et $\sigma_2 = \mathsf{snd}(\sigma)$, ceci équivaut à

$$\text{pour tout } M, N \in \mathcal{F}_{\mathrm{pub}}[\mathrm{dom}(\sigma_i)], \quad M\sigma_1 =_E N\sigma_1 \Leftrightarrow M\sigma_2 =_E N\sigma_2$$

Nous consacrons l'essentiel des chapitres 5 et 6 à une comparaison entre cette relation, appelée équivalence statique et la notion cryptographique d'indistinguabilité, en fonction de la théorie équationnelle E utilisée et de l'implémentation concrète des symboles de fonctions.

3.3.2 Correction et complétude

Afin de faire le lien entre les sémantiques normale et ouverte, nous commençons par définir l'ensemble des *expressions d'un processus* (et par extension d'un multi-ensemble de processus) de la manière suivante :

$$\begin{aligned}
\mathrm{expr}(0) &= \emptyset \\
\mathrm{expr}(P \mid Q) &= \mathrm{expr}(P) \cup \mathrm{expr}(Q) \\
\mathrm{expr}(!\, P) &= \mathrm{expr}(P) \\
\mathrm{expr}(u(x).\, P) &= \mathrm{expr}(P) \cup \{u\} \\
\mathrm{expr}(\overline{u}\langle v\rangle.\, P) &= \{u, v\} \cup \mathrm{expr}(P) \\
\mathrm{expr}(\text{if } u = v \text{ then } P \text{ else } Q) &= \{u, v\} \cup \mathrm{expr}(P) \cup \mathrm{expr}(Q) \\
\mathrm{expr}(\text{new } x.\, P) &= \mathrm{expr}(P) \\
\mathrm{expr}(\text{wait }.P) &= \mathrm{expr}(P)
\end{aligned}$$

Soit $\mathcal{K}_1 = \mathcal{E}_1; \mathcal{P}_{\mathrm{e},1}; \mathcal{P}_{\mathrm{a},1}$ et $\mathcal{K}_2 = \mathcal{E}_2; \mathcal{P}_{\mathrm{e},2}; \mathcal{P}_{\mathrm{a},2}$ deux configurations doubles. On note $\mathcal{K}_1 \preccurlyeq_n^{\mathrm{pub}} \mathcal{K}_2$ s'il existe un remplacement injectif ρ allant d'un sous-ensemble de $\mathcal{E}_1$ dans l'ensemble des constantes libres publiques n'apparaissant pas dans $\mathcal{K}_1$, tel que $\mathcal{E}_2 = \mathcal{E}_1 - \mathrm{dom}(\rho)$, $\mathcal{P}_{\mathrm{e}2} = \mathcal{P}_{\mathrm{e}1}\rho$ et $\mathcal{P}_{\mathrm{a}2} = \mathcal{P}_{\mathrm{a}1}\rho$. Notons que $\preccurlyeq_n^{\mathrm{pub}}$ est transitive.

Nous pouvons maintenant énoncer le lien entre les sémantiques normale et ouverte à l'aide des deux lemmes techniques suivants.

Lemme 3.8 (Complétude locale)**.** *Soit $\mathcal{K}$ une configuration double. Supposons qu'il existe deux pré-configurations doubles $\mathcal{K}_1 = \mathcal{E}_1; \mathcal{P}_{e,1}; \mathcal{P}_{a,1}$, $\mathcal{K}_2 = \emptyset; \mathcal{P}_{e,2}; \mathcal{P}_{a,2}$ et un ensemble de noms doubles $\mathcal{S}$ tels que*

– $\mathcal{K} \preccurlyeq_n^{pub} \mathcal{K}_1|\mathcal{K}_2$,
– $\mathrm{expr}(\mathcal{K}_2) \subseteq \mathcal{F}_{pub}[\mathcal{X}^1 \cup \mathcal{S}]$.

On a alors les implications suivantes :

(1) Si $\mathcal{K} \uparrow$ *alors* $\mathcal{K}_1; \mathcal{S} \uparrow_o$.

(2) Si $\mathcal{K} \to \mathcal{K}'$ *alors il existe deux pré-configurations doubles* $\mathcal{K}'_1 = \mathcal{E}'_1; \mathcal{P}'_{e,1}; \mathcal{P}'_{a,1}$, $\mathcal{K}'_2 = \emptyset; \mathcal{P}'_{e,2}; \mathcal{P}'_{a,2}$ *et un ensemble* $\mathcal{S}'$ *tels que*
– $\mathcal{K}' \preccurlyeq_n^{pub} \mathcal{K}'_1|\mathcal{K}'_2$,
– $\mathrm{expr}(\mathcal{K}'_2) \subseteq \mathcal{F}_{pub}[\mathcal{X}^1 \cup \mathcal{S}']$,
– $\mathcal{K}_1; \mathcal{S} \to_o^* \mathcal{K}'_1; \mathcal{S}'$.

Démonstration. (1) Tout d'abord, notons que si $\mathcal{K} \uparrow$ alors $(\mathcal{K}_1 \,|\, \mathcal{K}_2) \uparrow$. On distingue plusieurs cas suivant laquelle des deux règles de la table 3.6, (div-comm) ou (div-test), implique $\mathcal{K}_1 \,|\, \mathcal{K}_2 \uparrow$ et, selon son positionnement dans $\mathcal{K}_1 \,|\, \mathcal{K}_2$:
– Dans le cas de (div-test) ou (div-comm) à l'intérieur de $\mathcal{K}_1$, on obtient $\mathcal{K}_1; \mathcal{S} \uparrow_o$ par la même règle.
– Dans le cas où (div-comm) s'applique « à cheval » entre $\mathcal{K}_1$ et $\mathcal{K}_2$, c'est-à-dire que $u(x).P$ et $\overline{v}\langle t\rangle.Q$ sont des processus en cours d'exécution l'un dans $\mathcal{K}_1$, l'autre dans $\mathcal{K}_2$, alors, sachant que $\mathrm{expr}(\mathcal{K}_2) \subseteq \mathcal{F}_{\mathrm{pub}}[\mathcal{X}^1 \cup \mathcal{S}]$ (et que $\mathcal{K}_2$ est clos), on obtient $\mathcal{K}_1; \mathcal{S} \uparrow_o$ par la règle (div-recv) ou (div-send).
– Enfin, dans le cas de (div-test) ou (div-comm) à l'intérieur de $\mathcal{K}_2$, sachant que $\mathrm{expr}(\mathcal{K}_2) \subseteq \mathcal{F}_{\mathrm{pub}}[\mathcal{X}^1 \cup \mathcal{S}]$, on obtient $\mathcal{K}_1; \mathcal{S} \uparrow_o$ par la règle (div-static).

(2) Supposons $\mathcal{K} \to \mathcal{K}'$. Par un raisonnement identique à la preuve de la proposition 3.2, $\mathcal{K} \preccurlyeq_n^{\mathrm{pub}} \mathcal{K}_1|\mathcal{K}_2$ implique $\mathcal{K}_1|\mathcal{K}_2 \to \mathcal{K}''$ pour un certain $\mathcal{K}''$ tel que $\mathcal{K}' \preccurlyeq_n^{\mathrm{pub}} \mathcal{K}''$. Nous montrons qu'il existe deux configurations $\mathcal{K}'_1$ et $\mathcal{K}'_2 = \emptyset; \mathcal{P}'_{\mathrm{e},2}; \mathcal{P}'_{\mathrm{a},2}$, et un ensemble $\mathcal{S}'$ tels que
– $\mathcal{K}'' \preccurlyeq_n^{\mathrm{pub}} \mathcal{K}'_1|\mathcal{K}'_2$,
– $\mathrm{expr}(\mathcal{K}'_2) \subseteq \mathcal{F}_{\mathrm{pub}}[\mathcal{X}^1 \cup \mathcal{S}']$ et
– $\mathcal{K}_1; \mathcal{S} \to_o^* \mathcal{K}'_1; \mathcal{S}'$ (en fait en au plus une étape),
en distinguant plusieurs cas suivant la règle de la table 3.5 utilisée et son positionnement dans $\mathcal{K}_1$ et $\mathcal{K}_2$:
– Dans le cas d'une transition $\mathcal{K}_1|\mathcal{K}_2 \to \mathcal{K}'' = \mathcal{K}'_1|\mathcal{K}_2$ ayant lieu dans $\mathcal{K}_1$, on prend simplement $\mathcal{S}' = \mathcal{S}$, $\mathcal{K}'_2 = \mathcal{K}_2$.
– De même, dans le cas d'une transition $\mathcal{K}_1|\mathcal{K}_2 \to \mathcal{K}'' = \mathcal{K}_1|\mathcal{K}'_2$ différente de (new) ayant lieu dans $\mathcal{K}_2$, on prend simplement $\mathcal{S}' = \mathcal{S}$, $\mathcal{K}'_1 = \mathcal{K}_1$. Par inspection des règles, on a bien $\mathrm{expr}(\mathcal{K}'_2) \subseteq \mathcal{F}_{\mathrm{pub}}[\mathcal{X}^1 \cup \mathcal{S}^1]$.
– Dans le cas d'une règle (new) appliquée dans $\mathcal{K}_2$: $\mathcal{K}_1|\mathcal{K}_2 \to \mathcal{K}'' = \mathcal{K}_1|\mathcal{K}''_2$, on définit $\mathcal{K}'_2$ en remplaçant le nom $n \in \mathrm{names}(\mathcal{K}''_2) - \mathrm{names}(\mathcal{K}_1)$ créé dans $\mathcal{K}''_2$ par une constante publique libre fraîche, de sorte que $\mathcal{K}'' \preccurlyeq_n^{\mathrm{pub}} \mathcal{K}_1|\mathcal{K}'_2$ et la première composante de $\mathcal{K}'_2$ est $\emptyset$; puis l'on choisit $\mathcal{S}' = \mathcal{S}$ et $\mathcal{K}'_1 = \mathcal{K}_1$.
– Dans le cas de la règle (comm) appliquée entre un processus de $\mathcal{K}_1$ et un processus de $\mathcal{K}_2$, sachant que $\mathrm{expr}(\mathcal{K}_2) \subseteq \mathcal{F}_{\mathrm{pub}}[\mathcal{X}^1 \cup \mathcal{S}]$ et que les deux processus sont clos, on utilise l'une des deux règles (recv) ou (send). □

Lemme 3.9 (Correction locale)**.** *Soit* $\mathcal{I} = \mathcal{K}; \mathcal{S}$ *une configuration double étendue et* σ *une substitution partielle close à valeurs dans les termes doubles, telle que* $\mathcal{S} \subseteq \mathcal{F}_{pub}[\mathrm{im}(\sigma)]$, i.e. *tous les éléments de* $\mathcal{S}$ *sont engendrés par l'image de* σ.

(1) Si $\mathcal{I} \uparrow_o$, *alors il existe un processus simple* P *tel que* $\mathrm{var}(P) \subseteq \mathrm{dom}(\sigma)$,

$\mathrm{expr}(P\sigma) \subseteq \mathcal{F}_{pub}[\mathcal{X}^1 \cup \mathcal{S}]$,

$$\mathcal{K} \mid P\sigma \Uparrow$$

(2) Si $\mathcal{I} \to_o \mathcal{I}'$ *pour une certaine configuration double étendue* $\mathcal{I}' = \mathcal{K}'; \mathcal{S}'$*, alors il existe une substitution partielle close double* σ' *telle que*
- $\mathcal{S}' \subseteq \mathcal{F}_{pub}[\mathrm{im}(\sigma')]$,
- *pour tout processus simple* P' *vérifiant* $\mathrm{var}(P') \subseteq \mathrm{dom}(\sigma')$ *et* $\mathrm{expr}(P'\sigma') \subseteq \mathcal{F}_{pub}[\mathcal{X}^1 \cup \mathcal{S}']$*, il existe un processus simple* P *tel que* $\mathrm{var}(P) \subseteq \mathrm{dom}(\sigma)$, $\mathrm{expr}(P\sigma) \subseteq \mathcal{F}_{pub}[\mathcal{X}^1 \cup \mathcal{S}]$,

$$\mathcal{K} \mid P\sigma \to^* \mathcal{K}' \mid P'\sigma'$$

Démonstration. (1) On distingue plusieurs cas selon la règle impliquant $\mathcal{K}; \mathcal{S} \uparrow_o$.
- Dans le cas des règles (div-comm) et (div-test), on prend simplement $P = 0$.
- Dans le cas de (div-recv), avec les notations de la règle, soit $M \in \mathcal{F}_{\mathrm{pub}}[\mathrm{dom}(\sigma)]$ tel que $M\sigma = v$; on prend $P = \mathrm{new}\, x.\overline{M}\langle x\rangle$, de sorte la règle (div-comm) s'applique sur $\mathcal{K} \mid \emptyset; \{P\sigma\}; \emptyset$ après une réduction.
- Dans le cas de (div-send), avec les notations de la règle, soit $M \in \mathcal{F}_{\mathrm{pub}}[\mathrm{dom}(\sigma)]$ tel que $M\sigma = v$; on prend $P = M(x).0$ avec $\mathrm{sort}(x) = \mathrm{sort}(t)$; par conséquent la règle (div-comm) s'applique.
- Dans le cas de (div-static), avec les notations de la règle, soit $M, N \in \mathcal{F}_{\mathrm{pub}}[\mathrm{dom}(\sigma)]$ tel que $M\sigma = u$ et $N\sigma = v$; on prend

$$P = (\mathrm{if}\, M = N\ \mathrm{then}\, 0\ \mathrm{else}\, 0),$$

de sorte que la règle (div-test) s'applique.

(2) On distingue plusieurs cas selon la règle utilisée pour la transition $\mathcal{K}; \mathcal{S} \to_o \mathcal{K}'; \mathcal{S}'$.
- Le cas des règles autres que (wait), (recv) et (send) est clair : on prend $\sigma' = \sigma$ et pour tout processus P' vérifiant les conditions du lemme, $P = P'$.
- Concernant la règle (wait), on prend $\sigma' = \sigma$ et pour tout processus P', $P = \mathrm{wait}\, .P'$.
- Dans le cas de (recv), on prend $\sigma' = \sigma$; avec les notations de la règle, soit $M, N \in \mathcal{F}_{\mathrm{pub}}[\mathrm{dom}(\sigma)]$ tels que $M\sigma = t$ et $N\sigma = v$; pour tout processus P', on prend $P = \overline{N}\langle M\rangle.P'$.
- Dans le cas de (send), avec les notations de la règle, soit $x \notin \mathrm{dom}(\sigma)$ de même sorte que t ; on choisit $\sigma' = \sigma \uplus \{x \mapsto t\}$; soit $M \in \mathcal{F}_{\mathrm{pub}}[\mathrm{dom}(\sigma)]$ tel que $M\sigma = v$; pour tout processus P', on prend $P = M(x).P'$.

□

Application aux propriétés de sécurité. Parmi les propriétés de sécurité que nous avons mentionnées jusqu'à présent, nous avons vu (lemme 3.7) que la plupart s'énoncent en terme d'absence de divergence d'un certain bi-processus P^* : pour toute configuration publique simple $\mathcal{L}$, on demande $\mathcal{L} \mid P^* \not\Uparrow$.

À l'aide des lemmes de complétude et de correction locales (lemmes 3.8 et 3.9), nous transcrivons maintenant ces propriétés en terme de sûreté (inaccessibilité ou *safety*) pour la sémantique ouverte.

Théorème 3.10. *Soit* $\mathcal{K}$ *une configuration double. Les deux propositions suivantes sont équivalentes :*

(i) il existe une configuration publique simple $\mathcal{L}$ *telle que* $\mathcal{L} \mid \mathcal{K} \Uparrow$ *;*

(ii) la configuration double étendue $\mathcal{I}(\mathcal{K}) = \mathcal{K}; \emptyset$ *est telle que* $\mathcal{I}(\mathcal{K}) \Uparrow_o$.

Démonstration. On prouve cette proposition par récurrence sur la longueur de la séquence de transitions, à l'aide des deux lemmes techniques précédents.

- (i)⇒(ii) s'établit par le lemme de complétude locale : initialement $\mathcal{K}_1 = \mathcal{K}$, $\mathcal{K}_2 = \mathcal{L}$, $\mathcal{S} = \emptyset$;
- (ii)⇒(i) s'établit par le lemme de correction locale : initialement σ est la substitution partielle vide, $\mathcal{I} = \mathcal{I}(\mathcal{K})$; comme conclusion de la récurrence, on obtient l'existence d'un processus P tel que $\mathrm{var}(P) = \emptyset$, $\mathrm{expr}(P) \subseteq \mathcal{F}_{\mathrm{pub}}[\mathcal{X}_1]$ (en particulier P est donc public et simple) et $\mathcal{K} \mid P \Uparrow$. □

Nous terminons cette section par la preuve du lemme technique suivant utilisée pour prouver la proposition 3.6 de la section 3.2.

Lemme 3.11. *Soit* $P = C[Q]$ *un processus clos sans* wait *et* t *un terme tel que* $\mathrm{var}(t) \subseteq \mathrm{scope}(C)$. *Soit* $x \notin \mathrm{var}(t)$ *et*

$$P^* \quad \triangleq \quad C[\, Q \mid \mathrm{wait} \,.\, \mathrm{new}\, x.\, \overline{\mathsf{c}}\langle \mathrm{diff}[t, x] \rangle \,]$$

Si $\mathsf{fst}(P^*) \not\approx_{diff} \mathsf{snd}(P^*)$, *alors il existe un processus simple public* L *tel que* $L|P^* \Uparrow$ *et*

$$L = C_L[\mathrm{wait}\,.\mathsf{c}(x_1).\mathsf{c}(x_2) \ldots \mathsf{c}(x_n).\, \mathrm{if}\ u = v\ \mathrm{then}\, 0\, \mathrm{else}\, 0]$$

où C_L *est un contexte construit uniquement à partir des instructions* $u(x)$ *et* $\overline{u}\langle v \rangle$.

Démonstration. Par le théorème 3.10, $\mathsf{fst}(P^*) \not\approx_{\mathrm{diff}} \mathsf{snd}(P^*)$ implique $\mathcal{I}(P^*) \Uparrow_o$. Étant donné la forme de P et la sémantique du wait, la preuve de la réciproque du théorème (ii)⇒(i), à l'aide du lemme 3.9 montre l'existence d'un contexte L de la forme voulue. (Le n est dû aux réplications potentielles dans C.) □

3.4 Sémantique symbolique

La sémantique ouverte définie plus haut permet d'énoncer un nombre important de propriétés de sécurité comme des problèmes de sûreté (inaccessibilité des états néfastes). Toutefois, le système de transitions correspondant reste à branchement infini, notamment en ce qui concerne les messages t envoyés par l'attaquant (règle (recv) de la table 3.7) dont la taille est non bornée a priori.

Pour cette raison, il est courant de recourir à un troisième type de sémantiques, dites *sémantiques symboliques* [Hui99b, MS01, MS03, CKRT03a, DJ04a]. Ces sémantiques se caractérisent par le fait que lors de l'exécution d'un processus les messages émis par l'attaquant sont remplacés par des variables fraîches. Pour chaque trace d'exécution (symbolique) possible, on regroupe dans un système de contraintes les conditions de constructibilité des messages de l'attaquant ainsi que les tests opérés par les participants honnêtes.

3.4.1 Systèmes de contraintes symboliques

Nous décrivons maintenant les systèmes de contraintes sur lesquels repose la sémantique symbolique de notre langage.

Notations

Soit $\mathcal{X}^2$ un nouvel ensemble de variables dites *du second ordre*, notées $X, Y \ldots$, munies chacune d'une arité $\mathrm{ar}(X) \in \mathbb{N}$ et d'une sorte $\mathrm{sort}(X) \in \mathcal{T}$. On suppose qu'un nombre infini de variables X est disponible pour chaque arité et pour chaque sorte. Dans la suite, les variables de $\mathcal{X}^1$, encore notées $x, y \ldots$, sont appelées variables du *premier ordre*. On note $\mathcal{X} \triangleq \mathcal{X}^1 \cup \mathcal{X}^2$.

On étend la signature d'un ensemble

$$\mathcal{W} = \bigcup_{\tau \in \mathcal{T}} \{w_1^\tau, w_2^\tau, \ldots, w_k^\tau, \ldots\}$$

disjoint de $\mathcal{F}$, formé de constantes w_k^τ appelées *paramètres*, et vérifiant $\mathrm{sort}(w_k^\tau) = \tau$. L'entier $k > 0$ est appelé le *rang* de w_k^τ. On note encore E la théorie équationnelle obtenue en étendant la théorie précédente aux termes de $\mathcal{F}[\mathcal{W} \cup \mathcal{X}]$ de sorte que les paramètres sont des symboles libres dans E.

Dans la suite, nous réservons l'appellation *terme du premier ordre* aux éléments de $\mathcal{F}[\mathcal{X}^1]$, que l'on note $u, v, s, t \ldots$ Nous appelons (abusivement) *termes du second ordre* les éléments $M, N \ldots$ de $\mathcal{F}_{\mathrm{pub}}[\mathcal{W} \cup \mathcal{X}^2]$ contenant au plus un paramètre de chaque rang. Notons que les termes du second ordre ne contiennent pas de noms ni de symboles privés.

Étant donné un terme quelconque $T \in \mathcal{F}[\mathcal{W} \cup \mathcal{X}]$, on note $\mathrm{var}(T)$ l'ensemble des variables de T, $\mathrm{par}(T)$ l'ensemble des paramètres apparaissant dans T, $\mathrm{maxpar}(T)$ le rang maximal de ces paramètres, et $\mathrm{maxar}(T)$ l'arité maximale des variables du second ordre de T :

$$\begin{aligned} \mathrm{par}(T) &\triangleq \mathrm{st}(T) \cap \mathcal{W} \\ \mathrm{maxpar}(T) &\triangleq \max\{k \mid w_k^\tau \in \mathrm{par}(T)\} \\ \mathrm{maxar}(T) &\triangleq \max\{\mathrm{ar}(X) \mid X \in \mathrm{var}(T)\} \end{aligned}$$

avec la convention usuelle $\max(\emptyset) = -\infty$. Nous utilisons également les abréviations suivantes :

$$\begin{aligned} \mathrm{var}^1(T) &\triangleq \mathrm{var}(T) \cap \mathcal{X}^1 \\ \mathrm{var}^2(T) &\triangleq \mathrm{var}(T) \cap \mathcal{X}^2 \end{aligned}$$

Toutes ces notations ($\mathrm{var}(.)$, $\mathrm{var}^1(.)$, $\mathrm{var}^2(.)$, $\mathrm{par}(.)$, $\mathrm{maxpar}(.)$ et $\mathrm{maxar}(.)$) sont étendues aux ensembles ou aux n-uplets de termes de manière évidente.

Étant donné un terme T tel que $\mathrm{par}(T) \subseteq \{w_1^{\tau_1}, \ldots, w_m^{\tau_m}\}$, et m termes $T_1, \ldots, T_m$ de sortes respectives $\tau_1, \ldots, \tau_m$, on note $T[T_1, \ldots, T_m]$ le résultat du remplacement des $w_i^{\tau_i}$ par T_i dans T :

$$T[T_1, \ldots, T_m] \triangleq T\{w_1^{\tau_1} \mapsto T_1, \ldots, w_m^{\tau_m} \mapsto T_m\}.$$

Un *contexte public* C (sous-entendu clos mais non-nécessairement linéaire) est un terme du second ordre clos, autrement dit un terme $C \in \mathcal{F}_{\mathrm{pub}}[w_1^{\tau_1}, \ldots, w_m^{\tau_m}]$ pour certains $\tau_1, \ldots, \tau_m \in \mathcal{T}$.

Une substitution partielle σ à domaine inclus dans $\mathcal{X} = \mathcal{X}^1 \cup \mathcal{X}^2$ est *bien formée* ssi pour tout $x \in \mathrm{dom}(\sigma)$, $x\sigma$ est un terme du premier ordre de même sorte que x,

et que pour tout $X \in \mathrm{dom}(\sigma)$, $X\sigma$ est un terme du second ordre de même sorte que X, vérifiant

$$\mathrm{maxpar}(X\sigma) \leq \mathrm{ar}(X) \text{ et } \mathrm{maxar}(X\sigma) \leq \mathrm{ar}(X),$$

autrement dit, $\forall w_k^\tau \in \mathrm{par}(T)$, $k \leq \mathrm{ar}(X)$ et $\forall Y \in \mathrm{var}(T)$, $\mathrm{ar}(Y) \leq \mathrm{ar}(X)$.

On appelle *renommage bien formé inversible* un renommage bien formé ρ tel que ρ^{-1} est bien formé, c'est-à-dire tel que pour tout $X \in \mathrm{dom}(\rho)$, $\mathrm{ar}(X\rho) = \mathrm{ar}(X)$.

Dans la suite, sauf mention explicite du contraire, nous considérons uniquement des substitutions et des renommages bien formés.

Discussion sur le formalisme utilisé

Dans cette section et le chapitre suivant, nous adoptons un formalisme du second ordre sans λ-abstraction, inspiré des travaux sur l'équivalence statique [AF01, AC06], ou encore du codage employé par A. Degtyarev et A. Voronkov [DV96] pour réduire l'unification du second ordre à l'E-unification rigide simultanée.

En particulier, nos termes dits du second ordre utilisent des constantes spéciales w_k^τ appelées paramètres pour représenter les variables liées d'un λ-terme :

$$\lambda x_1, \ldots, x_n.t \;\approx\; M = t\{x_1 \mapsto w_1^{\tau_1}, \ldots, x_n \mapsto w_n^{\tau_n}\}$$

où $\tau_i = \mathrm{sort}(x_i)$. L'application d'arguments à un λ-terme correspond alors à des opérations explicites de remplacement des $w_i^{\tau_i}$:

$$(\lambda x_1, \ldots, x_n.t)(t_1, \ldots, t_n) \;\approx\; M[t_1, \ldots, t_n]$$

Dans un cadre cryptographique, les termes du second ordre représentent les calculs (ou *recettes* [AC06]) de l'attaquant. Ceci explique d'ores et déjà la restriction des termes M aux symboles publics, *i.e.* ceux disponibles à l'attaquant.

En outre, les termes M sont destinés à être appliqués aux messages connus de l'attaquant. Ces messages étant organisés par ordre chronologique et fixés pour chaque trace d'un protocole, on constate que dans un problème d'ordre cryptographique, les termes M s'appliquent à des préfixes d'un même n-uplet $(t_1, \ldots, t_n)$.

Ainsi, dans ce cadre, l'application peut être vue comme une opération uniforme de remplacement, envoyant chaque $w_i^{\tau_i}$ sur t_i (où $\tau_i = \mathrm{sort}(t_i)$). Cette simplification constitue un avantage notable du formalisme adopté.

Toujours dans un contexte cryptographique, les variables du second ordre X sont destinées à être instanciées par les calculs de l'attaquant. Du fait de la remarque précédente, il est commode de considérer l'arité des variables X comme un entier $\mathrm{ar}(X) = k$ plutôt que comme une signature $\tau_1 \times \cdots \times \tau_k \to \tau$, sachant qu'en pratique les τ_i seront fixés par le problème et que τ est redondant avec la notation $\mathrm{sort}(X)$.

Dans ces conditions, la contrainte de bonne formation des substitutions s'interprète comme une propriété de préservation du typage des λ-termes sous-jacents.

Systèmes de contraintes de l'intrus

Nous décrivons maintenant la syntaxe et la sémantique des systèmes de contraintes utilisés. Soit $\rhd$, $\rhd^?$, $=^?$, $=_E^?$ et $\neq_E^?$ des symboles binaires (en notation infixe).

Définition 3.12. Soit E une théorie équationnelle. Un *système de contraintes de l'intrus* sur E est un couple $\Sigma = \Phi; \mathcal{C}$ tel que

- Φ est une suite de termes du premier ordre, représentée sous forme d'un ensemble
$$\Phi = \{w_1^{\tau_1} \rhd t_1, \ldots, w_n^{\tau_n} \rhd t_n\}$$
où pour tout $1 \leq i \leq n$, $\tau_i = \mathrm{sort}(t_i)$, et l'on note $|\Phi| = n = \mathrm{maxpar}(\Phi)$;
- $\mathcal{C}$ est un ensemble de contraintes de la forme $X \rhd^? x$, où les variables X et x sont de même sorte et $\mathrm{ar}(X) \leq n$, ou bien de la forme $s =_E^? s'$ ou $s \neq_E^? s'$, où s, s' désignent des termes du premier ordre de même sorte ;

et tel que les conditions suivantes, dites *de régularité*, sont réalisées :

(1) pour tout $x \in \mathrm{var}(\mathcal{C})$, il existe un unique X tel que $(X \rhd^? x) \in \mathcal{C}$, et réciproquement chaque variable X apparaît au plus une fois dans $\mathcal{C}$;

(2) pour tous $1 \leq k \leq |\Phi|$ et $x \in \mathrm{var}(t_k)$, il existe une contrainte $(X \rhd^? x) \in \mathcal{C}$ telle que $\mathrm{ar}(X) < k$.

Une *solution* d'un tel système Σ est une substitution partielle close bien formée θ de domaine $\mathrm{dom}(\theta) = \mathrm{var}^2(\Sigma)$ telle qu'il existe une substitution partielle close bien formée λ de domaine $\mathrm{dom}(\lambda) = \mathrm{var}^1(\Sigma)$ vérifiant les conditions suivantes :

(i) pour tout $X \rhd^? x$ dans $\mathcal{C}$, $(X\theta)[t_1\lambda, \ldots, t_{\mathrm{ar}(X)}\lambda] = x\lambda$;

(ii) pour $s =_E^? s'$ dans $\mathcal{C}$, $s_j\lambda =_E s'_j\lambda$;

(iii) pour $s \neq_E^? s'$ dans $\mathcal{C}$, $s_j\lambda \neq_E s'_j\lambda$.

La substitution partielle λ est appelée *solution au premier ordre de* Σ *associée à* θ.

On appelle *structure de* Σ la donnée de $\mathrm{var}^2(\Sigma)$ et de $\mathrm{par}(\Sigma)$, c'est-à-dire la donnée de $\mathrm{var}^2(\Sigma)$ et des types $\tau_1, \ldots, \tau_n$. Un système de contraintes de l'intrus Σ est *positif* s'il ne comporte pas de contraintes de différence $s \neq_E^? s'$.

Notons que les conditions de régularité impliquent que pour toute solution éventuelle θ, il y a au plus une solution au premier ordre λ associée à θ. En effet, ces conditions (1) et (2), la condition (i) et la donnée de θ déterminent uniquement les valeurs $x\lambda$ par induction sur $\mathrm{ar}(X)$ où $(X \rhd^? x) \in \mathcal{C}$. Étant donnée une procédure de décision pour le problème du mot sur E, il est donc immédiat de vérifier si un certain θ est une solution de Σ.

Définition 3.13. L'ensemble des solutions d'un système de contraintes de l'intrus Σ est noté $\mathrm{sol}(\Sigma)$. Un système Σ est *satisfiable* ssi $\mathrm{sol}(\Sigma) \neq \emptyset$. Il est *universel* ssi $\mathrm{sol}(\Sigma)$ est l'ensemble des substitutions closes de domaine $\mathrm{var}^2(\Sigma)$.

Deux systèmes Σ_1 et Σ_2 ayant la même structure sont *équivalents* ssi $\mathrm{sol}(\Sigma_1) = \mathrm{sol}(\Sigma_2)$. Posons $\Sigma_i = \Phi_i; \mathcal{C}_i$. On dit que Σ_1 et Σ_2 sont *S-équivalents* ssi pour toute sorte $\tau \in \mathcal{T}$, étant donné des variables $X, x, Y, y \notin \mathrm{var}(\Sigma_1, \Sigma_2)$ de sortes τ telles que $\mathrm{ar}(X) = \mathrm{ar}(Y) = \mathrm{maxpar}(\Sigma_i)$, les systèmes de contraintes

$$\begin{array}{rcl} \Sigma'_1 & = & \Phi_1; \mathcal{C}_1 \cup \{X \rhd^? x,\ Y \rhd^? y,\ x =_E^? y\} \\ \text{et}\quad \Sigma'_2 & = & \Phi_2; \mathcal{C}_2 \cup \{X \rhd^? x,\ Y \rhd^? y,\ x =_E^? y\} \end{array} \tag{3.4.1}$$

sont équivalents. (Naturellement, cette dernière équivalence ne dépend pas du choix des variables X, x, Y, y de type τ.)

Notons que les notions de solution et donc d'(S-)équivalence se réfèrent uniquement aux valeurs des variables du second ordre. Ce point est crucial pour l'application à l'exécution symbolique de bi-processus que nous développons plus loin.

Exemple 3.6. Considérons à nouveau l'exemple simple du protocole Handshake :

$$\begin{array}{lll} 0. & A \to B : & \{N\}_{k_{AB}} \\ 1. & B \to A : & \{f(N)\}_{k_{AB}} \end{array}$$

ainsi que la signature (sans sorte) et la théorie équationnelle décrites précédemment (exemple 3.1) :

$$\mathsf{dec}(\mathsf{enc}(x,y),y) \doteq x \qquad \mathsf{enc}(\mathsf{dec}(x,y),y) \doteq x$$

Nous pouvons modéliser la faisabilité de la session principale de ce protocole « à la main » par le système $\Sigma = \Phi;\, \mathcal{C}$ suivant :

$$\begin{array}{rcl} \Phi & = & \{w_1 \rhd \mathsf{enc}(n,\mathsf{k}), \quad w_2 \rhd \mathsf{enc}(\mathsf{f}(\mathsf{dec}(x_1,\mathsf{k})),\mathsf{k})\,\} \\ \mathcal{C} & = & \{X_1 \rhd^? x_1, \quad X_2 \rhd^? x_2, \quad \mathsf{dec}(x_2,\mathsf{k}) =_E^? \mathsf{f}(n)\} \end{array}$$

où $\mathrm{ar}(X_1) = 1$, $\mathrm{ar}(X_2) = 2$, $n \in \mathcal{N}$ et k est une constante privée.

Σ admet pour solution la substitution correspondant au cours normal du protocole : $\theta = \{X_1 \mapsto w_1,\, X_2 \mapsto w_2\}$. En effet, le λ correspondant vérifie :

$$\begin{array}{rcl} x_1\lambda & = & (X_1\theta)[\mathsf{enc}(n,\mathsf{k})] = \mathsf{enc}(n,\mathsf{k}) \\ x_2\lambda & = & (X_2\theta)[\mathsf{enc}(n,\mathsf{k}),\mathsf{enc}(\mathsf{f}(\mathsf{dec}(x_1\lambda,\mathsf{k})),\mathsf{k})] \\ & = & \mathsf{enc}(\mathsf{f}(\mathsf{dec}(\mathsf{enc}(n,\mathsf{k}),\mathsf{k})),\mathsf{k}) \\ & =_E & \mathsf{enc}(\mathsf{f}(n),\mathsf{k}) \\ \mathsf{dec}(x_2,\mathsf{k})\lambda & =_E & \mathsf{f}(n) \end{array}$$

Le système Σ est donc satisfiable.

Exemple 3.7. Reprenons les notations ci-dessus. Soit $\Sigma_1 = \Phi_1;\, \mathcal{C}$ et $\Sigma_2 = \Phi_2;\, \mathcal{C}$ les systèmes définis par

$$\begin{array}{rcl} \Phi_1 & = & \Phi \cup \{w_3 \rhd \mathsf{k}\} \\ \Phi_2 & = & \Phi \cup \{w_3 \rhd \mathsf{k}'\} \end{array}$$

où k' est une autre constante privée.

Dans la mesure où $\mathrm{var}^2(\Sigma_i)$ ne contient pas de variables d'arité 3 ou plus, Σ_1 et Σ_2 sont clairement équivalents. Toutefois, ces deux systèmes ne sont pas S-équivalents. En effet, les systèmes Σ_1' et Σ_2' correspondants s'écrivent :

$$\begin{array}{rcl} \Sigma_1' & = & \Phi_1;\, \mathcal{C} \cup \{X \rhd^? x,\, Y \rhd^? y,\, x =_E^? y\} \\ \Sigma_2' & = & \Phi_2;\, \mathcal{C} \cup \{X \rhd^? x,\, Y \rhd^? y,\, x =_E^? y\} \end{array}$$

où $\mathrm{ar}(X) = \mathrm{ar}(Y) = 3$.

Soit $\theta' = \theta \uplus \{X \mapsto \mathsf{f}(\mathsf{dec}(w_1,w_3)), \quad Y \mapsto \mathsf{dec}(w_2,w_3)\}$. Dans le cas de Σ_1', le λ' correspondant vérifie

$$\begin{array}{rcl} x_1\lambda' & = & x_1\lambda \\ x_2\lambda' & = & x_2\lambda \\ x\lambda' & = & (X\theta)[\mathsf{enc}(n,\mathsf{k}),\mathsf{enc}(\mathsf{f}(\mathsf{dec}(x_1\lambda',\mathsf{k})),\mathsf{k}),\mathsf{k}] \;=_E\; \mathsf{f}(n) \\ y\lambda' & = & (Y\theta)[\mathsf{enc}(n,\mathsf{k}),\mathsf{enc}(\mathsf{f}(\mathsf{dec}(x_1\lambda',\mathsf{k})),\mathsf{k}),\mathsf{k}] \;=_E\; \mathsf{f}(n) \\ x\lambda' & =_E & y\lambda' \end{array}$$

donc $\theta \in \mathrm{sol}(\Sigma'_1)$. Or, dans le cas de Σ'_1, le λ' correspondant vérifie

$$\begin{array}{rcl} x_1\lambda' & = & x_1\lambda \\ x_2\lambda' & = & x_2\lambda \\ x\lambda' & = & (X\theta)[\mathsf{enc}(n,\mathsf{k}), \mathsf{enc}(\mathsf{f}(\mathsf{dec}(x_1\lambda',\mathsf{k})),\mathsf{k}),\mathsf{k}'] =_E \mathsf{f}(\mathsf{enc}(\mathsf{dec}(n,\mathsf{k}),\mathsf{k}')) \\ y\lambda' & = & (Y\theta)[\mathsf{enc}(n,\mathsf{k}), \mathsf{enc}(\mathsf{f}(\mathsf{dec}(x_1\lambda',\mathsf{k})),\mathsf{k}),\mathsf{k}'] =_E \mathsf{dec}(\mathsf{enc}(\mathsf{f}(n),\mathsf{k}),\mathsf{k}') \\ x\lambda' & \neq_E & y\lambda' \end{array}$$

d'où $\theta \notin \mathrm{sol}(\Sigma'_2)$. Σ'_1 et Σ'_2 ne sont donc pas équivalents ; et par conséquent Σ_1 et Σ_2 ne sont pas S-équivalents.

Cet exemple correspond intuitivement à l'attaque par dictionnaire déjà présentée dans l'exemple 3.5 : Σ_1 modélise le scénario d'une clé devinée correctement, tandis que Σ_2 modélise le scénario d'une clé incorrecte.

Nous verrons dans les sous-sections suivantes comment générer de tels systèmes de contraintes de manière automatique à partir du processus décrivant un protocole, et en accord avec les propriétés de sécurité définies précédemment.

Exemple 3.8. On remarque que l'exemple (simple) précédent utilise seulement un adversaire passif.

D'une manière générale, le cas d'un adversaire passif se ramène à deux systèmes *clos* $\Sigma_1 = \Phi_1; \emptyset$ et $\Sigma_2 = \Phi_2; \emptyset$ (une fois instanciée la substitution partielle correspondant au court normal du protocole, par ex. $\theta = \{X_1 \mapsto w_1, X_2 \mapsto w_2\}$ ci-dessus). De tels systèmes sont S-équivalents ssi les systèmes Σ'_1 et Σ'_2 suivants sont équivalents :

$$\begin{array}{rcl} \Sigma'_1 & = & \Phi_1; \{X \rhd^? x, Y \rhd^? y, x =^?_E y\} \\ \Sigma'_2 & = & \Phi_2; \{X \rhd^? x, Y \rhd^? y, x =^?_E y\} \end{array}$$

où $\mathrm{var}(\Phi_1) = \mathrm{var}(\Phi_2) = \emptyset$ et $\mathrm{ar}(X) = \mathrm{ar}(Y) = \mathrm{maxpar}(\Phi_1) = \mathrm{maxpar}(\Phi_2)$.

Posons $\Phi_i = \{w_1^{\tau_1} \rhd t_1^i, \ldots, w_n^{\tau_n} \rhd t_n^i\}$. Par définition des solutions de Σ'_1 et de Σ'_2 et sachant que les t_j^i sont clos, Σ_1 et Σ_2 sont S-équivalents ssi pour tous termes publics M et N tels que $\mathrm{par}(M), \mathrm{par}(N) \subseteq \mathrm{par}(\Phi_i)$, on a

$$M[t_1^1, \ldots, t_n^1] =_E N[t_1^1, \ldots, t_n^1] \iff M[t_1^2, \ldots, t_n^2] =_E N[t_1^2, \ldots, t_n^2]$$

Il s'agit à nouveau de la relation d'équivalence statique [AF01] discutée précédemment (sous-section 3.3.1) et qui sera utilisée de manière intensive dans les chapitres 5 et 6.

On peut donc voir la S-équivalence comme une généralisation de l'équivalence statique au cas d'un adversaire actif.

La proposition suivante montre différentes relations entre les problèmes précédents sur les systèmes de contraintes de l'intrus.

Proposition 3.12. *Étant donné un système de contraintes de l'intrus Σ, on note $\top(\Sigma)$ le problème universel obtenu à partir de Σ en supprimant toutes les contraintes $s =^?_E s'$ et $s \neq^?_E s'$. De plus, on note $\bot(\Sigma)$ le problème insatisfiable obtenu à partir de Σ en ajoutant une contrainte $\mathsf{c}_0 =^?_E \mathsf{c}_1$, où c_0 et c_1 sont deux constantes différentes modulo E.*

Les propriétés suivantes sont vérifiées pour tous Σ, Σ_1, Σ_2 :

(1) Si Σ_1 et Σ_2 sont S-équivalents, alors ils sont équivalents.

(2) Σ est universel ssi les deux systèmes Σ et $\top(\Sigma)$ sont équivalents, ou encore ssi Σ et $\top(\Sigma)$ sont S-équivalents.

(3) Σ est satisfiable ssi les deux systèmes Σ et $\bot(\Sigma)$ ne sont pas équivalents, ou encore ssi Σ et $\bot(\Sigma)$ ne sont pas S-équivalents.

(4) Supposons qu'il existe une sorte non dégénérée τ_0 ainsi que pour tout $\tau \in \mathcal{T} - \{\tau_0\}$ un symbole libre public $h_\tau : \tau \to \tau_0$.
Alors, Σ_1 et Σ_2 sont S-équivalents ssi les systèmes Σ'_1 et Σ'_2 sont équivalents, où Σ'_1 et Σ'_2 sont définis comme ci-dessus (équation 3.4.1) pour des variables X, x, Y, y de sorte τ_0.

Démonstration. (1) Posons $\Sigma_i = \Phi_i; \mathcal{C}_i$ pour $i \in \{1, 2\}$. Soit $\Sigma'_i = \Phi_i; \mathcal{C}_i \cup \{X \rhd^? x, Y \rhd^? y, x =^?_E y\}$, défini comme ci-dessus pour des variables X, x, Y, y d'un type quelconque τ. Soit θ une solution de Σ_1 (par exemple). Soit c une constante de type τ. $\theta' = \theta \uplus \{X \mapsto \mathsf{c}, Y \mapsto \mathsf{c}\}$ est une solution de Σ'_1, donc également de Σ'_2 par hypothèse. Vu la forme de Σ'_2, on en déduit que θ est solution de Σ_2.

(2) Sachant que $\top(\Sigma)$ est universel et de même structure que Σ, on a bien l'équivalence : Σ est universel ssi Σ et $\top(\Sigma)$ sont équivalents. D'après le point (1), il reste à vérifier que si $\top(\Sigma)$ et Σ sont équivalents alors ils sont également S-équivalents.

Soit $\top(\Sigma)'$ et Σ' des systèmes obtenus comme ci-dessus par l'ajout de contraintes $X \rhd^? x$, $Y \rhd^? y$, $x =^?_E y$ dans $\top(\Sigma)$ et Σ respectivement.

Dans la mesure où $\top(\Sigma)'$ s'obtient à partir de Σ' en enlevant des contraintes d'égalité ou de différence, toute solution de Σ' est solution de $\top(\Sigma)'$.

Inversement, soit θ' une solution de $\top(\Sigma)'$, et λ' la solution au premier ordre associée. La substitution partielle $\theta = \theta'|_{\mathrm{var}^2(\Sigma)}$ est solution de $\top(\Sigma)$, donc également de Σ par hypothèse. Sachant que les systèmes $\top(\Sigma)$ et λ ne diffèrent que par les contraintes d'égalité et de différence. θ admet la même solution au premier ordre λ dans $\top(\Sigma)$ et Σ. On en déduit que λ' est également solution au premier ordre de θ' dans Σ'.

(3) Soit $\bot(\Sigma)'$ et Σ' des systèmes obtenus comme ci-dessus par l'ajout de contraintes $X \rhd^? x$, $Y \rhd^? y$, $x =^?_E y$ dans $\bot(\Sigma)$ et Σ respectivement.

Du fait de la contrainte $\mathsf{c}_0 =^?_E \mathsf{c}_1$, $\bot(\Sigma)$ et $\bot(\Sigma)'$ sont insatisfiables. De plus, par raisonnement similaire à (1), Σ est satisfiable ssi Σ' l'est.

(4) La propriété est évidente compte tenu de l'existence des symboles publics libres $h_\tau : \tau \to \tau_0$ ($\tau \neq \tau_0$) et du corollaire 2.3. □

Dans le chapitre 4, nous étudions le problème de la satisfiabilité et la S-équivalence de tels systèmes pour une classe générale de théories équationnelles, dites sous-terme-convergentes (définies au chapitre 2).

Formulation en terme d'E-unification du second ordre

Dans une perspective plus générale et pour éclairer (peut-être) la définition précédente, on peut également voir les systèmes de contraintes de l'intrus comme un type particulier de problèmes d'E-(dis)unification du second ordre [Dow01, OND98].

En effet, soit $\Sigma = \Phi; \mathcal{C}$ un système de contraintes de l'intrus. Supposons pour simplifier que $\mathrm{maxar}(\mathcal{C}) = \mathrm{maxpar}(\Phi)$, hypothèse à laquelle on peut toujours se ramener en ajoutant à $\mathcal{C}$ des contraintes $X \rhd^? x$ inutiles (c'est-à-dire telles que X et x n'apparaissent nulle part ailleurs dans Σ).

Soit $X_1, \ldots, X_m$ les variables du second ordre de Σ et $a_i = \mathrm{ar}(X_i)$. Supposons que $0 \leq a_1 \leq a_2 \leq \ldots \leq a_m = n$, $\Phi = \{w_1^{\tau_1} \rhd t_1, \ldots, w_n^{\tau_n} \rhd t_n\}$ et

$$\mathcal{C} = \{X_1 \rhd^? x_1, \ldots, X_m \rhd^? x_m,\ s_1 \sim_1 s'_1, \ldots, s_q \sim_n s'_q\}$$

où $\sim_j \in \{=^?_E, \neq^?_E\}$. Les contraintes de Σ s'écrivent alors sous la forme suivante :

$$\exists x_1, \ldots, x_m,\quad \left\{\begin{array}{lcl} X_1[t_1, \ldots, t_{a_1}] & =^? & x_1 \\ & \ldots & \\ X_m[t_1, \ldots, t_{a_m}] & =^? & x_m \end{array}\right. \quad \text{et} \quad \left\{\begin{array}{lcl} s_1 & \sim_1 & s'_1 \\ & \ldots & \\ s_q & \sim_q & s'_q \end{array}\right.$$

tandis que les conditions de régularité deviennent :

(1) $\mathrm{var}(t_1, \ldots, t_{a_m}, s_1, \ldots, s_q, s'_1, \ldots, s'_q) \subseteq \{x_1, \ldots, x_m\}$ et les variables X_1, …, X_m, x_1, …, x_m sont deux à deux distinctes ;

(2) pour tout $1 \leq i \leq m$ et $1 \leq j \leq a_i$, $\mathrm{var}(t_j) \subseteq \{x_1, \ldots, x_{i-1}\}$.

La quantification existentielle $\exists x_1, \ldots, x_m$ indique que les valeurs des variables du premier ordre ne sont pas prises en compte dans la notion de solution. Notons que les conditions sur les occurrences de ces variables permettent de les éliminer du système en substituant x_i par $X_i[t_1, \ldots, t_{a_i}]$, ou alternativement de considérer toutes les égalités modulo E.

La forme ci-dessus est celle d'un problème d'E-unification du second ordre «classique » [Dow01, OND98], à ceci près que nous autorisons a priori des contraintes de différence modulo E et que nous donnons la possibilité d'interdire certains symboles dans la solution des X_i (en l'occurrence les symboles de $\mathcal{F} - \mathcal{F}_{\mathrm{pub}} = \mathcal{F}_{\mathrm{priv}} \cup \mathcal{N}$). Cette dernière possibilité est cruciale pour les applications cryptographiques, où les variables X_i représentent les opérations réalisables par l'attaquant, lequel ignore a priori les secrets du protocole.

3.4.2 Définition de la sémantique symbolique

Forts de la notion de système de contraintes de l'intrus, nous pouvons maintenant donner une sémantique symbolique aux processus et plus généralement aux configurations doubles.

Appelons *système de contraintes de l'intrus double* un système de contraintes de la forme précédente (définition 3.12 et conditions relatives) excepté que les termes t_k, s, s' sont des termes doubles. Un tel système double Σ décrit donc deux systèmes de contraintes de l'intrus simples $\mathsf{fst}(\Sigma)$ et $\mathsf{snd}(\Sigma)$ — a fortiori ayant la même structure.

Définition 3.14. Une *configuration symbolique double* est un quintuplet $\mathcal{J} = \mathcal{E}; \mathcal{P}_{\mathrm{e}}; \mathcal{P}_{\mathrm{a}}; \Phi; \mathcal{C}$ où

- $\mathcal{E}$ est un ensemble de noms ;
- $\mathcal{P}_{\mathrm{e}}$ et $\mathcal{P}_{\mathrm{a}}$ sont des multi-ensembles de processus doubles ;
- $\Phi; \mathcal{C}$ est un système de contraintes de l'intrus double ;

tel que les conditions suivantes soient vérifiées :

(1) $\mathrm{names}(\mathcal{P}_\mathrm{e}, \mathcal{P}_\mathrm{a}, \Phi, \mathcal{C}) \subseteq \mathcal{E}$;

(2) pour tout $x \in \mathrm{var}(\mathcal{J})$, il existe une unique variable X telle que $(X \rhd^? x) \in \mathcal{C}$.

Une configuration double $\mathcal{K} = \mathcal{E}; \mathcal{P}_\mathrm{e}; \mathcal{P}_\mathrm{a}$ est vue comme une configuration double symbolique

$$\mathcal{J}(\mathcal{K}) \triangleq \mathcal{E}; \mathcal{P}_\mathrm{e}; \mathcal{P}_\mathrm{a}; \emptyset; \emptyset.$$

On définit la relation de transition entre configurations symboliques $\rightarrow_s$ et le prédicat de divergence $\uparrow_s$ correspondant par les règles des table 3.9 et 3.10. Comme précédemment, $\Uparrow_s$ désigne le prédicat $\rightarrow_s^* \uparrow_s$.

Les règles (nil), (par), (repl), (new), (wait) et (next) étendent simplement les règles de la sémantique ouverte. La règle (comm) permet l'échange d'un message entre deux processus pourvu que les sortes correspondantes soient compatibles ; l'égalité des canaux d'émission et de réception se traduit par une nouvelle contrainte $u =_E^? v$. De même, (test-1) et (test-2) permettent d'exécuter un test en ajoutant la contrainte d'égalité ou de différence correspondante, selon la branche choisie. La règle (recv) correspond à la réception d'un message émis par l'intrus ; la sémantique étant symbolique, le message reçu est représenté par une variable fraîche y ; les nouvelles contraintes traduisent le fait que le canal u et le message y doivent être constructibles par l'intrus. La règle (send) permet l'envoi d'un message t à destination de l'intrus ; ce message est ajouté à la suite Φ des messages reçus par l'intrus ; la nouvelle contrainte traduit le fait que le canal u est constructible.

Enfin, la règle (div-Σ) définit les configurations divergentes comme étant celles pour lesquelles les deux systèmes de contraintes associés à la configuration ne sont pas S-équivalents.

3.4.3 Correction et complétude

Étant données une configuration double étendue $\mathcal{I}$ et une configuration symbolique double $\mathcal{J} = \mathcal{E}; \mathcal{P}_\mathrm{e}; \mathcal{P}_\mathrm{a}; \Phi; \mathcal{C}$ avec $\Phi = \{w_1^{\tau_1} \rhd t_1, \ldots, w_n^{\tau_n} \rhd t_n\}$ et $\Sigma = \Phi; \mathcal{C}$, on note $\mathcal{I} \preccurlyeq \mathcal{J}$ lorsqu'il existe une substitution partielle close (simple) θ et une substitution partielle close double λ vérifiant les conditions suivantes :

(i) $\mathcal{I}$ est de la forme $\mathcal{I} = \mathcal{E}; \mathcal{P}_\mathrm{e}\lambda; \mathcal{P}_\mathrm{a}\lambda; \mathcal{S}$ où $\mathcal{S} = \{t_1\lambda, \ldots, t_n\lambda\}$;

(ii) $\mathsf{fst}(\lambda)$ et $\mathsf{snd}(\lambda)$ sont solutions au premier ordre associées à θ pour les systèmes $\mathsf{fst}(\Sigma)$ et $\mathsf{snd}(\Sigma)$ respectivement.

Les deux lemmes techniques qui suivent établissent le lien entre sémantiques ouverte et symbolique.

Lemme 3.13 (Complétude locale)**.** *Soit $\mathcal{I}$ une configuration double étendue et $\mathcal{J}$ une configuration symbolique double telles que $\mathcal{I} \preccurlyeq \mathcal{J}$.*

On a les implications suivantes :

(1) Si $\mathcal{I} \uparrow_o$ alors $\mathcal{J} \Uparrow_s$.

(2) Si $\mathcal{I} \rightarrow_o \mathcal{I}'$ alors il existe une configuration symbolique double $\mathcal{J}'$ telle que $\mathcal{J} \rightarrow_s \mathcal{J}'$ et $\mathcal{I}' \preccurlyeq \mathcal{J}'$.

Démonstration. Soit θ et λ vérifiant les conditions (i) et (ii) justifiant la propriété $\mathcal{I} \preccurlyeq \mathcal{J}$. Soit $\mathcal{I} = \mathcal{E}; \mathcal{P}_\mathrm{e}\lambda; \mathcal{P}_\mathrm{a}\lambda; \mathcal{S}$, $\mathcal{J} = \mathcal{E}; \mathcal{P}_\mathrm{e}; \mathcal{P}_\mathrm{a}; \Phi; \mathcal{C}$ et $\Sigma = \Phi; \mathcal{C}$.

(1) Supposons $\mathcal{I} \uparrow_o$. On distingue plusieurs cas selon la règle applicable.

$$\mathcal{E};\ \{0\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \to_s \mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \qquad \text{(nil)}$$

$$\mathcal{E};\ \{P \,|\, Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \to_s \mathcal{E};\ \{P,\, Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \qquad \text{(par)}$$

$$\mathcal{E};\ \{!\,P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \to_s \mathcal{E};\ \{P,\, !\,P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \qquad \text{(repl)}$$

$$\left(\begin{array}{l}\mathcal{E};\ \{u(x).\,P,\ \bar{v}\langle t\rangle.\,Q\} \uplus \mathcal{P}_{\mathrm{e}};\\ \quad \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C}\end{array}\right) \to_s \left(\begin{array}{l}\mathcal{E};\ \{P\{x \mapsto t\},\, Q\} \uplus \mathcal{P}_{\mathrm{e}};\\ \quad \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \cup \{u =^?_E v\}\end{array}\right) \qquad \text{(comm)}$$
$$\text{si} \left\{\begin{array}{l}\mathrm{sort}(u) = \mathrm{sort}(v)\\ \mathrm{sort}(x) = \mathrm{sort}(t)\end{array}\right.$$

$$\left(\begin{array}{l}\mathcal{E};\ \{\text{if } u = v \text{ then } P \text{ else } Q\}\\ \quad \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C}\end{array}\right) \to_s \left(\begin{array}{l}\mathcal{E};\ \{P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\\ \quad \Phi;\ \mathcal{C} \cup \{u =^?_E v\}\end{array}\right) \qquad \text{(test-1)}$$

$$\left(\begin{array}{l}\mathcal{E};\ \{\text{if } u = v \text{ then } P \text{ else } Q\}\\ \quad \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C}\end{array}\right) \to_s \mathcal{E};\ \{Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \cup \{u \neq^?_E v\} \qquad \text{(test-2)}$$

$$\mathcal{E};\ \{\text{new } x.P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \to_s \{n\} \cup \mathcal{E};\ \{P\{x \mapsto n\}\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \qquad \text{(new)}$$
$$\text{si} \left\{\begin{array}{l}\mathrm{sort}(n) = \mathrm{sort}(x)\\ n \in \mathcal{N} - \mathcal{E}\end{array}\right.$$

$$\mathcal{E};\ \{\text{wait }.P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \to_s \mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \{P\} \uplus \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \qquad \text{(wait)}$$

$$\mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \to_s \mathcal{E};\ \mathcal{P}_{\mathrm{a}};\ \emptyset;\ \Phi;\ \mathcal{C} \qquad \text{si } \mathcal{P}_{\mathrm{a}} \neq \emptyset \qquad \text{(next)}$$

$$\mathcal{E};\ \{u(x).\,P\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \to_s \left(\begin{array}{l}\mathcal{E};\ \{P\{x \mapsto y\}\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\\ \mathcal{C} \cup \{Y \rhd^? y,\ Z \rhd^? z,\ z =^?_E u\}\end{array}\right) \qquad \text{(recv)}$$
$$\text{si} \left\{\begin{array}{l} y, Y, z, Z \notin \mathrm{var}(\mathcal{C})\\ \mathrm{ar}(Y) = \mathrm{ar}(Z) = |\Phi|\\ \mathrm{sort}(y) = \mathrm{sort}(Y) = \mathrm{sort}(x)\\ \mathrm{sort}(z) = \mathrm{sort}(Z) = \mathrm{sort}(u)\end{array}\right.$$

$$\mathcal{E};\ \{\bar{u}\langle t\rangle.\,Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \to_s \left(\begin{array}{l}\mathcal{E};\ \{Q\} \uplus \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi \cup \{w^\tau_n \rhd t\};\\ \quad \mathcal{C} \cup \{Z \rhd^? z,\ z =^?_E u\}\end{array}\right) \qquad \text{(send)}$$
$$\text{si} \left\{\begin{array}{l} z, Z \notin \mathrm{var}(\mathcal{C}),\ \mathrm{ar}(Z) = |\Phi|\\ n = |\Phi| + 1, \quad \tau = \mathrm{sort}(t)\\ \mathrm{sort}(z) = \mathrm{sort}(Z) = \mathrm{sort}(u)\end{array}\right.$$

Tab. 3.9 – Transitions entre configurations symboliques doubles

$$\mathcal{E};\ \mathcal{P}_{\mathrm{e}};\ \mathcal{P}_{\mathrm{a}};\ \Phi;\ \mathcal{C} \quad \uparrow_s \qquad \text{si } \Sigma = \Phi;\mathcal{C} \text{ est tel que } \mathsf{fst}(\Sigma) \text{ et } \mathsf{snd}(\Sigma) \text{ ne sont pas S-équivalents.} \qquad \text{(div-}\Sigma\text{)}$$

Tab. 3.10 – Configurations symboliques doubles divergentes

– Dans le cas de la règle (div-static), avec les notations de la règle, par définition il existe $u, v \in \mathcal{F}_{\text{pub}}[\mathcal{S}]$ tels que $u \uparrow_E v$, c'est-à-dire par exemple $\mathsf{fst}(u) =_E \mathsf{fst}(v)$ mais $\mathsf{snd}(u) \neq_E \mathsf{snd}(v)$.
Comme $u, v \in \mathcal{F}_{\text{pub}}[\mathcal{S}]$, et que Φ est de la forme $\{w_1^{\tau_1} \rhd t_1, \ldots, w_n^{\tau_n} \rhd t_n\}$ avec $\mathcal{S} = \{t_1\lambda, \ldots, t_n\lambda\}$, il existe $M, N \in \mathcal{F}_{\text{pub}}[w_1^{\tau_1}, \ldots, w_n^{\tau_n}]$ tels que

$$\begin{array}{rcl} M[\mathsf{fst}(t_1\lambda), \ldots, \mathsf{fst}(t_n\lambda)] & =_E & N[\mathsf{fst}(t_1\lambda), \ldots, \mathsf{fst}(t_n\lambda)] \quad \text{et} \\ M[\mathsf{snd}(t_1\lambda), \ldots, \mathsf{snd}(t_n\lambda)] & \neq_E & N[\mathsf{snd}(t_1\lambda), \ldots, \mathsf{snd}(t_n\lambda)]. \end{array}$$

Soit $X, x, Y, y \notin \text{var}(\mathcal{C})$ des variables de même sorte que u et v, où $\text{ar}(X) = \text{ar}(Y) = |\Phi|$. Soit Σ' le système double défini par $\Sigma' = \Phi; \mathcal{C} \cup \{X \rhd^? x,\ Y \rhd^? y,\ x =_E^? y\}$. Soit $\theta' = \theta \uplus \{X \mapsto M,\ Y \mapsto N\}$. θ' est une solution de $\mathsf{fst}(\Sigma')$ mais non de $\mathsf{snd}(\Sigma')$. Par conséquent, les deux systèmes $\mathsf{fst}(\Sigma)$ et $\mathsf{snd}(\Sigma)$ ne sont pas S-équivalents et la règle (div-Σ) s'applique.

– Si l'une des règles (div-comm), (div-test), (div-recv) ou (div-send) s'applique, alors on montre qu'il existe $\mathcal{J}'$ tel que $\mathcal{J} \rightarrow_s \mathcal{J}'$ par la règle correspondante, respectivement (comm), (test-1), (recv) ou (send), et tel que $\mathcal{J}' \uparrow_s$. En effet, considérons par exemple le cas de (div-recv) (les autres cas se traitent de manière similaire).
Comme (div-recv) s'applique à $\mathcal{I} = \mathcal{E}; \mathcal{P}_{\text{e}}\lambda; \mathcal{P}_{\text{a}}\lambda; \mathcal{S}$, $\mathcal{P}_{\text{e}}$ s'écrit alors $\mathcal{P}_{\text{e}} = \mathcal{P}_{\text{e0}} \uplus \{u(x).P\}$ avec $u\lambda \uparrow_E v$ pour un certain $v \in \mathcal{F}_{\text{pub}}[\mathcal{S}]$. On a donc

$$\mathcal{J} \rightarrow_s \mathcal{J}' = \mathcal{E}; \mathcal{P}_{\text{e0}} \cup \{P\{x \mapsto y\}\}; \mathcal{P}_{\text{a}}; \Phi; \mathcal{C} \cup \{Y \rhd^? y,\ Z \rhd^? z,\ z =_E^? u\}$$

par la règle (recv).
Comme Φ est de la forme $\{w_1^{\tau_1} \rhd t_1, \ldots, w_n^{\tau_n} \rhd t_n\}$ avec $\mathcal{S} = \{t_1\lambda, \ldots, t_n\lambda\}$, il existe $M \in \mathcal{F}_{\text{pub}}[w_1^{\tau_1}, \ldots, w_n^{\tau_n}]$ tels que

$$M[t_1\lambda, \ldots, t_n\lambda] = v.$$

Soit $\Sigma' = \Phi; \mathcal{C} \cup \{Y \rhd^? y,\ Z \rhd^? z,\ z =_E^? u\}$ et $\theta' = \theta \uplus \{Y \mapsto \mathsf{c},\ Z \mapsto M\}$ où c est une constante de même sorte que Y. Comme $u\lambda \uparrow_E v$, θ' est solution d'un des deux systèmes $\mathsf{fst}(\Sigma')$ et $\mathsf{snd}(\Sigma')$ mais non de l'autre.

(2) On distingue à nouveau plusieurs cas selon la règle appliquée dans la sémantique ouverte. Dans chaque cas, on vérifie que la règle correspondante de la sémantique symbolique aboutit à un $\mathcal{J}'$ convenable. Le cas des règles (nil), (par), (repl), (comm), (test-1), (test-2), (new), (wait), (next) est clair. Pour les règles (recv) et (send), on montre que la configuration $\mathcal{J}'$ obtenue vérifie $\mathcal{I}' \preccurlyeq \mathcal{J}'$ en construisant la solution θ' comme précédemment à partir de θ et des valeurs $v, t \in \mathcal{F}_{\text{pub}}[\mathcal{S}]$ de la règle non symbolique. □

Une configuration symbolique double $\mathcal{J} = \mathcal{E}; \mathcal{P}_{\text{e}}; \mathcal{P}_{\text{a}}; \Phi; \mathcal{C}$ est dite *cohérente* lorsque les deux systèmes $\mathsf{fst}(\Sigma)$ et $\mathsf{snd}(\Sigma)$, où $\Sigma = \Phi; \mathcal{C}$, sont équivalents, c'est-à-dire ont les mêmes ensembles de solutions (éventuellement vides).

Lemme 3.14 (Correction locale)**.** *Soit $\mathcal{J}$ une configuration symbolique double. On a les implications suivantes :*

(1) Si $\mathcal{J}$ est cohérente et $\mathcal{J} \uparrow_s$, alors il existe une configuration double étendue $\mathcal{I}$ telle que $\mathcal{I} \preccurlyeq \mathcal{J}$ et $\mathcal{I} \uparrow_o$.

(2) Si $\mathcal{J}$ est cohérente et $\mathcal{J} \to_s \mathcal{J}'$, alors alors l'une condition suivante au moins est vérifiée :

(a) il existe une configuration double étendue $\mathcal{I}$ telle que $\mathcal{I} \preccurlyeq \mathcal{J}$ et $\mathcal{I} \uparrow_o$

(b) $\mathcal{J}'$ est cohérente et pour tout $\mathcal{I}' \preccurlyeq \mathcal{J}'$, il existe $\mathcal{I}$ telle que $\mathcal{I} \preccurlyeq \mathcal{J}$ et $\mathcal{I} \to_o \mathcal{I}'$.

Démonstration. Posons $\mathcal{J} = \mathcal{E}; \mathcal{P}_\mathrm{e}; \mathcal{P}_\mathrm{a}; \Phi; \mathcal{C}$ et $\Sigma = \Phi; \mathcal{C}$.

(1) Supposons $\mathcal{J}$ cohérente et $\mathcal{J} \uparrow_s$. Il existe donc un type τ, des variables $X, Y, x, y \notin \mathrm{var}(\mathcal{C})$ de sorte τ et d'arité $\mathrm{ar}(X) = \mathrm{ar}(Y) = \mathrm{maxar}(\mathcal{C})$, et une substitution partielle close θ' tels que, étant donné $\Sigma' = \Phi; \mathcal{C} \cup \{X \rhd^? x,\ Y \rhd^? y,\ x =^?_E y\}$, θ' est (par exemple) solution de $\mathsf{fst}(\Sigma')$ mais non de $\mathsf{snd}(\Sigma')$.
Étant donné la forme de Σ', en particulier $\theta = \theta'|_{\mathrm{var}^2(\Sigma)}$ est solution de $\mathsf{fst}(\Sigma)$. Comme $\mathcal{J}$ est cohérent, θ est également solution de $\mathsf{snd}(\Sigma)$. Autrement dit, θ' est solution des deux projections du système double $\Phi; \mathcal{C} \cup \{X \rhd^? x,\ Y \rhd^? y\}$. Soit λ' la solution double au premier ordre associée à θ' dans ce dernier système. Notons que $\lambda = \lambda'|_{\mathrm{var}^1(\mathcal{C})}$ est la solution double au premier ordre associée à θ dans Σ.
Soit $\Phi = \{w_1^{\tau_1} \rhd t_1, \ldots, w_n^{\tau_n} \rhd t_n\}$. D'après ce qui précède et par définition des systèmes de contraintes de l'intrus, on a donc

$$\begin{aligned} (X\theta')[t_1\lambda, \ldots, t_n\lambda] &= x\lambda' \\ (Y\theta')[t_1\lambda, \ldots, t_n\lambda] &= y\lambda' \\ \mathsf{fst}(x\lambda') &=_E \mathsf{fst}(y\lambda') \\ \mathsf{snd}(x\lambda') &\neq_E \mathsf{snd}(y\lambda') \end{aligned}$$

Soit $\mathcal{I} = \mathcal{E}; \mathcal{P}_\mathrm{e}\lambda; \mathcal{P}_\mathrm{a}\lambda; \mathcal{S}$ où $\mathcal{S} = \{t_1\lambda, \ldots, t_n\lambda\}$. On a bien $\mathcal{I} \preccurlyeq \mathcal{J}$. Or, si l'on note $u = x\lambda'$ et $v = y\lambda'$, on a $u, v \in \mathcal{F}_\mathrm{pub}[\mathcal{S}]$ et $u \uparrow_E v$, donc $\mathcal{I} \uparrow_o$ par la règle (div-static).

(2) Supposons $\mathcal{J}$ cohérente et $\mathcal{J} \to_s \mathcal{J}'$. Soit $\mathcal{J}' = \mathcal{E}'; \mathcal{P}'_\mathrm{e}; \mathcal{P}'_\mathrm{a}; \Phi'; \mathcal{C}'$ et $\Sigma' = \Phi'; \mathcal{C}'$. On distingue plusieurs cas selon la règle utilisée.
- Dans le cas des règles (nil), (par), (new), (wait) et (next), $\Sigma' = \Sigma$ est a fortiori cohérent. Supposons $\mathcal{I}' \preccurlyeq \mathcal{J}'$ pour des substitutions θ et λ. Étant donné $\mathcal{I}' = \mathcal{E}'; \mathcal{P}'_\mathrm{e}\lambda; \mathcal{P}'_\mathrm{a}\lambda; \mathcal{S}$, on pose $\mathcal{I} = \mathcal{E}; \mathcal{P}_\mathrm{e}\lambda; \mathcal{P}_\mathrm{a}\lambda; \mathcal{S}$, de sorte que $\mathcal{I} \preccurlyeq \mathcal{J}$. De plus, on vérifie facilement que $\mathcal{I} \to_o \mathcal{I}'$ par la règle du même nom que la transition symbolique.
- Dans le cas des autres règles (comm), (test-1), (test-2), (recv), (send), on distingue encore deux cas suivant si les contraintes ajoutées au système rendent $\mathcal{J}'$ incohérent. Considérons en détail le cas de (recv). (Les autres règles se traitent de manière similaire.) Nous avons

$$\begin{aligned} \mathcal{J} &= \mathcal{E}; \mathcal{P}_{\mathrm{e}0} \uplus \{u(x).P\}; \mathcal{P}_\mathrm{a}; \Phi; \mathcal{C} \\ \mathcal{J}' &= \mathcal{E}; \mathcal{P}_{\mathrm{e}0} \uplus \{P\{x \mapsto y\}\}; \mathcal{P}_\mathrm{a}; \Phi; \mathcal{C} \cup \{Y \rhd^? y,\ Z \rhd^? z,\ z =^?_E u\} \end{aligned}$$

où y, Y, z, Z vérifient les contraintes énoncées dans la règle. Posons $\Phi = \{w_1^{\tau_1} \rhd t_1, \ldots, w_n^{\tau_n} \rhd t_n\}$.
- Si $\mathcal{J}'$ n'est pas cohérent, alors il existe (par exemple) une solution θ' de $\mathsf{fst}(\Sigma')$ qui n'est pas une solution de $\mathsf{snd}(\Sigma')$. Soit $\theta = \theta'|_{\mathrm{var}^2(\mathcal{C})}$. Comme Σ est cohérent, θ est une solution de $\mathsf{fst}(\Sigma)$ et $\mathsf{snd}(\Sigma)$, ou de manière

équivalente, θ' est solution des deux projections de Φ; $\mathcal{C} \cup \{Y \rhd^? y,\ Z \rhd^? z\}$. Soit λ' la solution double au premier ordre associée à θ' dans ce dernier système.
Par définition des systèmes de contraintes de l'intrus, on a donc

$$\begin{array}{rcl} (Z\theta')[t_1\lambda, \ldots, t_n\lambda] & = & z\lambda' \\ \mathsf{fst}(z\lambda') & =_E & \mathsf{fst}(u\lambda') \\ \mathsf{snd}(z\lambda') & \neq_E & \mathsf{snd}(u\lambda') \end{array}$$

Soit $\mathcal{I} = \mathcal{E}; \mathcal{P}_{\mathrm{e}}\lambda; \mathcal{P}_{\mathrm{a}}\lambda; \mathcal{S}$ où $\mathcal{S} = \{t_1\lambda, \ldots, t_n\lambda\}$. On a bien $\mathcal{I} \preccurlyeq \mathcal{J}$. Or, si l'on note $v = z\lambda'$, on a $v \in \mathcal{F}_{\mathrm{pub}}[\mathcal{S}]$ et $u\lambda' \uparrow_E v$, donc $\mathcal{I} \uparrow_o$ par la règle (div-recv).

– Sinon, $\mathcal{J}'$ est cohérent. Nous vérifions la seconde partie de la condition (b). Soit $\mathcal{I}'$ quelconque tel que $\mathcal{I}' \preccurlyeq \mathcal{J}'$. Soit θ' et λ' telles que :

$$\mathcal{I}' = \mathcal{E}; \mathcal{P}_{\mathrm{e}0}\lambda' \uplus \{P\{x \mapsto y\}\lambda'\}; \mathcal{P}_{\mathrm{a}}\lambda'; \mathcal{S}$$

où $\mathcal{S} = \{t_1\lambda', \ldots, t_n\lambda'\}$, et λ' est solution double au premier ordre associée à θ' pour le système double Σ'.
Soit $\theta = \theta'|_{\mathrm{var}^2(\mathcal{C})}$ et $\lambda = \lambda'|_{\mathrm{var}^2(\mathcal{C})}$. Dans la mesure où $y, Y, z, Z \notin \mathrm{var}(\mathcal{C})$, λ est la solution au premier ordre associée à θ dans $\Sigma = \Phi; \mathcal{C}$.
Soit $\mathcal{I} = \mathcal{E}; \mathcal{P}_{\mathrm{e}0}\lambda \uplus \{(u(x).P)\lambda\}; \mathcal{P}_{\mathrm{a}}\lambda; \mathcal{S}$. On a donc $\mathcal{I} \preccurlyeq \mathcal{J}$.
De plus, par définition des systèmes de contraintes, on a

$$\begin{array}{rcl} (Y\theta')[t_1\lambda, \ldots, t_n\lambda] & = & y\lambda' \\ (Z\theta')[t_1\lambda, \ldots, t_n\lambda] & = & z\lambda' \\ z\lambda' & =_E & u\lambda \end{array}$$

En prenant $t = y\lambda' \in \mathcal{F}_{\mathrm{pub}}[\mathcal{S}]$ et $v = z\lambda' \in \mathcal{F}_{\mathrm{pub}}[\mathcal{S}]$, on a donc $\mathcal{I} \rightarrow_o \mathcal{I}'$. □

Application à la diff-équivalence. Nous pouvons maintenant caractériser l'absence de divergence d'un bi-processus ainsi que les propriétés de sécurité qui en découlent en terme de sémantique symbolique.

Théorème 3.15. *Soit $\mathcal{K}$ une configuration double. Les propositions suivantes sont équivalentes :*

(i) il existe une configuration publique simple $\mathcal{L}$ telle que $\mathcal{L} \,|\, \mathcal{K} \Uparrow$;

(ii) la configuration double étendue $\mathcal{I}(\mathcal{K})$ est telle que $\mathcal{I}(\mathcal{K}) \Uparrow_o$;

(iii) la configuration symbolique double $\mathcal{J}(\mathcal{K})$ vérifie $\mathcal{J}(\mathcal{K}) \Uparrow_s$;

Démonstration. Nous avons montré (i)⇔(ii) dans le théorème 3.10.

(ii)⇒(iii). Supposons $\mathcal{I}(\mathcal{K}) \Uparrow_o$. Sachant que trivialement $\mathcal{I}(\mathcal{K}) \preccurlyeq \mathcal{J}(\mathcal{K})$, on obtient $\mathcal{J}(\mathcal{K}) \Uparrow_s$ grâce au lemme de complétude locale (lemme 3.13) par induction sur la longueur de la dérivation $\mathcal{I}(\mathcal{K}) \rightarrow_o^* \uparrow_o$.

(iii)⇒(ii). Supposons $\mathcal{J}(\mathcal{K}) \Uparrow_s$. Sachant que $\mathcal{J}(\mathcal{K})$ est trivialement cohérent (la seule solution d'un système vide est la substitution partielle vide), par induction sur la longueur de la dérivation $\mathcal{J}(\mathcal{K}) \rightarrow_s^* \uparrow_s$, le lemme de correction locale

(lemme 3.14) implique l'existence d'une configuration double étendue $\mathcal{I}$ telle que $\mathcal{I} \Uparrow_o$ et $\mathcal{I} \preccurlyeq \mathcal{J}(\mathcal{K})$. Or $\mathcal{J}(\mathcal{K}) = \emptyset; \{\mathcal{K}\}; \emptyset; \emptyset; \emptyset$. On a donc $\mathcal{I}(\mathcal{K}) = \mathcal{I} \Uparrow_o$. □

3.5 Décidabilité de la diff-équivalence pour les processus finis

Forts de la sémantique symbolique des processus, nous étudions maintenant le problème de la décidabilité de la diff-équivalence entre configurations finies. On appelle *fini* tout processus ou configuration (simple ou double) ne comportant pas d'instruction de réplication « ! ».

3.5.1 Réduction vers la S-équivalence (ou la satisfiabilité)

En absence de réplication, toutes les sémantiques considérées jusqu'à présent terminent fortement sur les configurations finies, dans la mesure où chaque transition consomme au moins une instruction. Toutefois les relations de transition entre configurations restent a priori à branchement infini.

Soit $\cong_{\mathrm{n}}$ la relation d'équivalence entre configurations symboliques doubles définie par : $\mathcal{J} \cong_{\mathrm{n}} \mathcal{J}'$ ss'il existe deux renommages (bijectifs) ρ_1 : names$(\mathcal{J}) \to$ names$(\mathcal{J}')$ et $\rho_2 : \mathrm{var}(\mathcal{J}) \to \mathrm{var}(\mathcal{J}')$ tels que $\mathcal{J}' = \mathcal{J}\rho_1\rho_2$.

Sachant que les noms sont libres dans E (définition 2.2), on montre facilement que la sémantique symbolique est compatible avec le renommage dans le sens suivant :

Lemme 3.16. *$\mathcal{J}_1 \cong_n \mathcal{J}_1'$ implique*

$$\mathcal{J}_1 \uparrow_s \Leftrightarrow \mathcal{J}_1' \uparrow_s$$
$$\forall \mathcal{J}_2, \quad \mathcal{J}_1 \to_s \mathcal{J}_2 \;\Rightarrow\; \exists \mathcal{J}_2',\ \mathcal{J}_1' \to_s \mathcal{J}_2' \text{ et } \mathcal{J}_2 \cong_n \mathcal{J}_2'$$

Ce lemme nous permet de considérer les configurations symboliques modulo $\cong_{\mathrm{n}}$. Dans ces conditions, la sémantique symbolique devient à branchement fini. Par le lemme de König et le théorème 3.15, la décision de la diff-équivalence se ramène donc à un nombre fini de problèmes de S-équivalence.

De manière plus précise, munissons les configurations de la notion de taille définie comme le nombre total d'instructions contenus dans chaque processus (codage des squelettes de processus sans partage) augmenté du nombre de sous-termes distincts dans l'ensemble des expressions de la configuration (codage des termes avec partage, les variables liées étant supposées distinctes). De même, munissons les problèmes de S-équivalences de la notion de taille correspondant aux nombres de sous-termes distincts dans les systèmes de contraintes en entrée.

Le théorème suivant donne la relation entre la diff-équivalence de bi-processus (et plus généralement configurations doubles) et la S-équivalence de systèmes de contraintes de l'intrus.

Théorème 3.17. *On a les faits suivants :*

1. *La diff-équivalence entre configurations finies est résoluble en temps polynomial non-déterministe avec oracle pour la S-équivalence.*

2. *La diff-équivalence entre configurations finies, positives (i.e. sans tests de différence, ni branche* else *autre que* 0*) est résoluble en temps polynomial non-déterministe avec oracle pour la S-équivalence entre systèmes positifs.*

Démonstration.

1. Nous avons déjà noté que chaque transition symbolique consomme au moins une instruction. Or, une telle transition introduit au plus 4 nouveaux sous-termes dans la configuration symbolique (pour la règle (recv)). Par induction, on obtient que la taille des systèmes de contraintes accessibles par la sémantique symbolique (au sens ci-dessus) est borné linéairement par la taille de la configuration initiale (toujours au sens ci-dessus).

   On conclut ce point par le théorème 3.15 en notant que, modulo renommage des noms, le nombre de transitions possibles à partir d'une configuration symbolique est borné par le nombre d'instructions de la configuration.

2. Nous vérifions que la règle (test-2) de la table 3.9 n'est pas nécessaire pour décider l'absence de divergence dans le cas des processus positifs, c'est-à-dire formellement :

   > (*) Pour toute configuration symbolique positive $\mathcal{J}$ telle que $\mathcal{J} \Uparrow_s$, il existe une réduction $\mathcal{J} \to_s^* \uparrow_s$ n'utilisant pas la règle (test-2).

   En effet, sans cette règle tous les systèmes de contraintes engendrés sont positifs. On peut alors conclure par un raisonnement identique au point (1).

   Notons $\to_{spos}$ la relation de transition symbolique sans règle (test-2), et soit $\Uparrow_{spos} \stackrel{\Delta}{=} (\to_{spos}^* \uparrow_s)$. Soit $\preceq$ la relation entre configurations symboliques définie comme la clôture réflexive et transitive de la relation suivante : $\mathcal{J}_1 \prec \mathcal{J}_2$ ssi $\mathcal{J}_1$ et $\mathcal{J}_2$ sont de la forme

   $$\begin{array}{rcl} \mathcal{J}_1 & = & \mathcal{E};\, \{0\} \uplus \mathcal{P}_{\mathrm{e}};\, \mathcal{P}_{\mathrm{a}};\, \Phi;\, \mathcal{C} \cup \{u \neq_E^? v\} \\ & \text{ou} & \mathcal{E};\, \mathcal{P}_{\mathrm{e}};\, \mathcal{P}_{\mathrm{a}};\, \Phi;\, \mathcal{C} \cup \{u \neq_E^? v\} \\ \mathcal{J}_2 & = & \mathcal{E};\, \{\text{if } u = v \text{ then } P \text{ else } 0\} \uplus \mathcal{P}_{\mathrm{e}}; \mathcal{P}_{\mathrm{a}}; \Phi;\, \mathcal{C} \end{array}$$

   Par analyse des règles, on vérifie facilement que pour toutes configurations positives $\mathcal{J}_1$ et $\mathcal{J}_2$ telles que $\mathcal{J}_1 \prec \mathcal{J}_2$,
   - $\mathcal{J}_1 \uparrow_s$ implique $\mathcal{J}_2 \uparrow_s$;
   - $\mathcal{J}_1 \to_s \mathcal{J}_1'$ implique qu'il existe $\mathcal{J}_2'$ telle que $\mathcal{J}_2 \to_{spos}^* \mathcal{J}_2'$ (en fait en au plus une étape) et $\mathcal{J}_1' \preceq \mathcal{J}_2'$, où $\mathcal{J}_1'$ et $\mathcal{J}_2'$ sont également positives.

   Par clôture transitive de $\prec$, ces deux propriétés sont également vraies lorsque $\mathcal{J}_1 \preceq \mathcal{J}_2$.

   Ainsi, par clôture transitive de $\to_s$, on obtient que pour toutes configurations positives $\mathcal{J}_1$ et $\mathcal{J}_2$ telles que $\mathcal{J}_1 \preceq \mathcal{J}_2$, $\mathcal{J}_1 \Uparrow_s$ implique $\mathcal{J}_2 \Uparrow_{spos}$. On en déduit la propriété voulue (*) en prenant $\mathcal{J}_1 = \mathcal{J}_2 = \mathcal{J}$. □

Du théorème précèdent, on déduit en particulier la décidabilité de l'existence des attaques par dictionnaire pour un nombre fini de sessions d'un protocole, en supposant donné un oracle pour la S-équivalence.

Considérons à nouveau le codage des propriétés d'accessibilité en terme de diff-équivalence (lemme 3.7) : étant donné une configuration simple $\mathcal{K}$ telle que les seules occurrences de la constante libre privée bad dans $\mathcal{K}$ sont dans le sous-processus $\overline{\mathsf{bad}}\langle\rangle$, nous avons montré que (i) $\mathcal{L} \mid \mathcal{K} \not\Downarrow_{\mathsf{bad}}$ pour toute configuration publique simple $\mathcal{L}$ ssi (ii) la configuration double $\mathcal{K}' = \mathcal{K}\{\mathsf{bad} \mapsto \mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]\}$ est sans divergence, où c_0, c_1 sont deux constantes publiques distinctes modulo E. Par le théorème 3.15, ceci équivaut à $\mathcal{J}(\mathcal{K}') \not\Downarrow_s$.

En étudiant les systèmes de contraintes engendrés par de tels problèmes d'accessibilité, on peut particulariser le théorème 3.17 de la manière suivante.

Théorème 3.18. *On a les faits suivants :*

1. *Les propriétés d'accessibilité (au sens ci-dessus) sur les configurations finies sont résolubles en temps polynomial non-déterministe avec oracle pour la satisfiabilité des systèmes de contraintes de l'intrus.*
2. *Les propriétés d'accessibilité sur les configurations finies, positives sont résolubles en temps polynomial non-déterministe avec oracle pour la satisfiabilité des systèmes positifs.*

Démonstration. Avec les notations ci-dessus, on peut supposer sans perte de généralité que les constantes c_0 et c_1 sont libres et n'apparaissent pas dans $\mathcal{K}'$. Étant donné la forme de $\mathcal{K}'$, par analyse des règles, les configurations symboliques atteignables à partir de $\mathcal{J}(\mathcal{K}')$ sont de la forme $\mathcal{J} = \mathcal{E}; \mathcal{P}_\mathrm{e}; \mathcal{P}_\mathrm{a}; \Phi; \mathcal{C}$ où le seule terme double (ou contenant c_0 ou c_1) apparaissant dans $\mathcal{J}$ est $\mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]$; de plus, cette expression apparaît uniquement aux endroits suivants :

- dans un sous-processus $\overline{\mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]}\langle\rangle$ de $\mathcal{P}_\mathrm{e}$ et $\mathcal{P}_\mathrm{a}$, ou
- dans une ou plusieurs règles $w_i^{\tau_i} \rhd \mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]$ de Φ, ou
- dans une équation $u =^?_E \mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]$ de $\mathcal{C}$, où le terme simple u ne contient pas c_0 ni c_1.

On vérifie facilement qu'une configuration symbolique $\mathcal{J}$ de cette forme est divergente ssi le terme double $\mathrm{diff}[\mathsf{c}_0, \mathsf{c}_1]$ apparaît dans Φ et que le système de contraintes $\mathsf{fst}(\Phi; \mathcal{C})$ est satisfiable. (Noter que $\mathsf{fst}(\Phi; \mathcal{C})$ est satisfiable ssi $\mathsf{snd}(\Phi; \mathcal{C}) = \mathsf{fst}(\Phi; \mathcal{C})\{\mathsf{c}_0 \mapsto \mathsf{c}_1\}$ l'est.)

Autrement dit, lorsque l'on applique le raisonnement de la preuve du théorème 3.17, tous les problèmes de S-équivalence engendrés par un problème d'accessibilité se réduisent à des problèmes de satisfiabilité (positifs dans le cas des configurations positives). □

Remarque. Le codage de l'authentification présenté dans la section 3.2 comporte des instructions de réplication dans les processus T_inj et $T_\mathrm{non\text{-}inj}$. Au sens strict, la condition P fini n'implique donc pas P^* fini.

Toutefois, on peut facilement borner le nombre de réplications nécessaires dans P^* en fonction du nombre maximal d'événements begin et end émis dans les sessions considérées. On considère donc le processus (fini) dans lequel les réplications de P^* sont dépliées un nombre suffisant de fois.

Ainsi, le théorème 3.18 implique la décidabilité du secret simple et de l'authentification pour un nombre fini de sessions d'un protocole, en supposant donné un oracle pour la satisfiabilité des systèmes de contraintes.

3.5.2 Bornes inférieures de complexité

Nous donnons brièvement quelques éléments permettant de minorer la classe de complexité des problèmes d'accessibilité (et donc de la diff-équivalence par le lemme 3.7) en fonction de la théorie équationnelle considérée.

Pour commencer, notons que l'on dispose d'une réciproque au théorème 3.18.

Proposition 3.19. *La satisfiabilité des systèmes de contraintes (respectivement des systèmes de contraintes positifs) se réduit en temps polynomial à l'accessibilité (respectivement sur les configurations positives).*

Démonstration. Soit c une constante publique et $\Sigma = \Phi;\mathcal{C}$ un système de contraintes de l'intrus. Il existe des types $\tau_1, \ldots, \tau_n$, des variables $X_1, \ldots, X_m$, $x_1, \ldots, x_m$, et des termes $t_1, \ldots, t_n$, $s_1, \ldots, s_q$, et $s'_1, \ldots, s'_q$ tels que :

$$\begin{array}{rcl} \Phi & = & \{w_1^{\tau_1} \rhd t_1, \ldots, w_n^{\tau_n} \rhd t_n\} \\ \mathcal{C} & = & \{X_1 \rhd^? x_1, \ldots, X_m \rhd^? x_m,\ s_1 \sim_1 s'_1, \ldots, s_q \sim_n s'_q\} \end{array}$$

où $\sim_j \in \{=^?_E, \neq^?_E\}$ et $0 \leq \mathrm{ar}(X_1) \leq \ldots \leq \mathrm{ar}(X_m) \leq n$.

Identifions provisoirement les symboles $=^?_E$ et $\neq^?_E$ avec $=$ et $\neq$ respectivement. Soit bad une constante libre privée n'apparaissant pas dans Σ. On considère le processus $P(\Sigma)$ défini de la manière suivante :

$$\begin{array}{rcl} P(\Sigma) & \triangleq & \mathsf{c}(x_{i_0+1}), \ldots, \mathsf{c}(x_{i_1}).\overline{\mathsf{c}}\langle t_1\rangle. \\ & & \mathsf{c}(x_{i_1+1}), \ldots, \mathsf{c}(x_{i_2}).\overline{\mathsf{c}}\langle t_2\rangle. \\ & & \ldots \\ & & \mathsf{c}(x_{i_{n-1}+1}), \ldots, \mathsf{c}(x_{i_n}).\overline{\mathsf{c}}\langle t_n\rangle. \\ & & \mathsf{c}(x_{i_n+1}), \ldots, \mathsf{c}(x_{i_{n+1}}). \\ & & \text{if } s_1 \sim_1 s'_1 \text{ and } \ldots \text{ and } s_q \sim_n s'_q \text{ then } \overline{\mathsf{bad}}\langle\rangle \end{array}$$

pourvu que

- $0 = i_0 \leq i_1 \leq \ldots \leq i_{n+1} = m$,
- pour tout $0 \leq k \leq n$, pour tout $i_k + 1 \leq j \leq i_k$, $\mathrm{ar}(X_j) = k$.

On vérifie facilement que $P(\Sigma) \rightarrow_s^* \downarrow_{\mathsf{bad}}$ ssi Σ est satisfiable. De plus, $P(\Sigma)$ est positif ssi Σ l'est. □

La construction utilisée dans cette dernière preuve ne se généralise pas immédiatement à la S-équivalence. Intuitivement, la différence de comportement entre processus et systèmes de contraintes réside dans le fait qu'un intrus peut utiliser hors-ligne les messages déjà envoyés par un processus, même si le processus est sur le point d'échouer. A contrario, deux systèmes de contraintes insatisfiables sont toujours S-équivalents. Pour cette raison, nous conjecturons que la S-équivalence se réduit à la diff-équivalence pour certaines théories équationnelles mais non en général.

Par ailleurs, on dispose d'une réduction évidente allant de l'E-unification vers la satisfiabilité des systèmes de contraintes.

Proposition 3.20. *Supposons comme précédemment que la signature comprend une nombre infini de constantes libres publiques de chaque sorte.*

Soit $u =^?_E v$ un problème d'E-unification tels que les termes du premier ordre u et v soient publics, i.e. $u, v \in \mathcal{F}_{pub}[\mathcal{X}^1]$.

Soit $x_1, \ldots, x_m$ les variables (deux à deux distinctes) apparaissant dans u et v. Soit $X_1, \ldots, X_m$ des variables deux à deux distinctes, d'arité 0 et de sortes appropriées. Alors l'équation $u =^?_E v$ est satisfiable ssi le système suivant l'est :

$$\Sigma_{u,v} = \emptyset;\ \{X_i \rhd^? x_i\}_{1 \leq i \leq m} \cup \{u =^?_E v\}$$

Démonstration. Sachant qu'un nombre infini de constantes publiques libres est disponible dans la signature, par renommage, $u =^?_E v$ est satisfiable ss'il existe une substitution partielle λ telle que $u\lambda =_E v\lambda$ et $\mathrm{im}(\lambda) \subseteq \mathcal{F}_{\mathrm{pub}}[\emptyset]$. Les contraintes de déductibilité sont alors satisfaites de manière triviale. □

Par les propositions 3.19 et 3.12 et le lemme 3.7 (sur le codage de l'accessibilité en terme de diff-équivalence), ce résultat implique en particulier l'indécidabilité de tous les problèmes considérés lorsque l'E-unification est indécidable (sur les symboles publics de la signature).

Pour conclure ce paragraphe, nous montrons que la satisfiabilité des systèmes de contraintes est NP-difficile dans le cas de la théorie vide (*i.e.* $=_E$ est l'égalité syntaxique), lorsque la signature, non typée, contient au moins un symbole binaire et une constante privée. En effet, la proposition suivante montre que l'on peut coder le problème 3-SAT dans ce problème.

Proposition 3.21. *Considérons une signature non sortée contenant un symbole binaire h et une constante privée k. Posons $c_0 = \mathsf{h}(\mathsf{k}, \mathsf{k})$, $c_1 = \mathsf{h}(c_0, \mathsf{k})$, $k_1 = \mathsf{h}(c_1, \mathsf{k})$ et $k_{p+1} = \mathsf{h}(k_p, \mathsf{k})$ pour tout entier $p \geq 1$.*

Soit m et n deux entiers positifs. Soit $X_1, \ldots, X_n$, $Y_1, \ldots, Y_m$ et $x_1, \ldots, x_n$, $y_1, \ldots, y_m$ des variables deux à deux distinctes, telles que $\mathrm{ar}(X_j) = 2$ et $\mathrm{ar}(Y_i) = 3m + 2$.

On suppose donnés des indices $j_{ik} \in \{1, \ldots, n\}$ et des termes $s_{ik} \in \{c_0, c_1\}$ pour tous $1 \leq i \leq m$ et $1 \leq k \leq 3$. Le terme s_{ik} représente le signe (par exemple c_1 : positif ou c_0 : négatif) de la variable $x_{j_{ik}}$ dans la i-ème clause d'une instance de 3-SAT :

$$\begin{array}{rl} & (\pm_{s_{11}} x_{j_{11}} \vee \ldots \vee \pm_{s_{13}} x_{j_{13}}) \\ \wedge & \ldots \\ \wedge & (\pm_{s_{m1}} x_{j_{m1}} \vee \ldots \vee \pm_{s_{m3}} x_{j_{m3}}) \end{array}$$

Considérons le système $\Sigma = \Phi; \mathcal{C}$ défini par

$$\begin{aligned} \Phi = \{ & w_1 \rhd c_0,\ w_2 \rhd c_1, \\ & w_3 \rhd \mathsf{h}(k_1, x_{j_{11}}),\ w_4 \rhd \mathsf{h}(\mathsf{h}(k_1, s_{11}), x_{j_{12}}),\ w_5 \rhd \mathsf{h}(\mathsf{h}(\mathsf{h}(k_1, s_{11}), s_{12}), x_{j_{13}}) \\ & \qquad \ldots \\ & w_{3m} \rhd \mathsf{h}(k_m, x_{j_{m1}}),\ w_{3m+1} \rhd \mathsf{h}(\mathsf{h}(k_m, s_{m1}), x_{j_{m2}}), \\ & \qquad w_{3m+2} \rhd \mathsf{h}(\mathsf{h}(\mathsf{h}(k_m, s_{m1}), s_{m2}), x_{j_{m3}}) \ \} \end{aligned}$$

$$\begin{aligned} \mathcal{C} = \{ & X_1 \rhd^? x_1,\ \ldots,\ X_n \rhd^? x_n,\ Y_1 \rhd^? y_1,\ \ldots,\ Y_m \rhd^? y_m, \\ & y_1 =^? \mathsf{h}(\mathsf{h}(\mathsf{h}(k_1, s_{11}), s_{12}), s_{13}),\ \ldots,\ y_m =^? \mathsf{h}(\mathsf{h}(\mathsf{h}(k_m, s_{m1}), s_{m2}), s_{m3}) \ \} \end{aligned}$$

Alors, une substitution partielle close (bien formée) λ est solution au premier ordre de Σ ssi $\lambda|_{x_1,\ldots,x_n}$ est solution au sens habituel de la formule logique :

$$\begin{array}{ll} & \left(x_{j_{11}} =^? s_{11} \vee \ldots \vee x_{j_{13}} =^? s_{13}\right) \\ \wedge & \ldots \\ \wedge & \left(x_{j_{m1}} =^? s_{m1} \vee \ldots \vee x_{j_{m3}} =^? s_{m3}\right) \end{array}$$

Schéma de preuve. Dans la mesure où la constante privée k n'est déductible d'aucune instance de Φ, ni utilisée comme deuxième argument de h dans les termes du système, les termes $c_0, c_1, k_1, k_2 \ldots$ se comportent comme des constantes libres privées distinctes deux à deux.

Sachant que les termes c_0 et c_1 sont donnés dans Φ, on observe que le terme $\mathsf{h}(\mathsf{h}(\mathsf{h}(k_i, s_{i1}), s_{i2}), s_{i3})$ est constructible à partir de Φ ssi l'un des sous-termes $\mathsf{h}(\mathsf{h}(\mathsf{h}(k_i, s_{i1}), s_{i2}), s_{i3})$, $\mathsf{h}(\mathsf{h}(k_i, s_{i1}), s_{i2})$, $\mathsf{h}(k_i, s_{i1})$ s'unifie avec l'un des termes de Φ, en l'occurrence correspondant à w_{3i}, w_{3i+1} ou w_{3i+2}. □

Dans la mesure où l'unification syntaxique est de complexité polynomiale, ceci montre qu'en général, la satisfiabilité des systèmes de contraintes de l'intrus ne se réduit pas polynomialement à l'E-unification (et de même a fortiori pour tous les problèmes auxquels elle se réduit).

3.6 Conclusion

Ainsi, les théorèmes 3.17 et 3.18 impliquent la décidabilité de toutes les propriétés de sécurité du chapitre 3.2 énoncées en terme d'(in)accessibilité ou de diff-équivalence, pour un nombre borné de sessions, moyennant des oracles pour les problèmes correspondants sur les systèmes de contraintes. Les propriétés concernées incluent notamment

- le secret simple,
- l'authentification,
- le secret (simple) par anticipation (*forward secrecy*), et
- l'absence d'attaques par dictionnaire.

Pour le secret fort fondé sur une relation d'équivalence plus générale que $\cong_{\text{diff}}$, la diff-équivalence donne également une approximation correcte d'après le théorème 3.4.

Nous avons également observé que tous les problèmes considérés encodent la satisfiabilité des systèmes de contraintes de l'intrus (lemme 3.7 et propositions 3.12, 3.19) et par conséquent l'E-unification (proposition 3.20). Ceci implique mécaniquement l'indécidabilité de ces propriétés pour toutes les théories E pour lesquelles l'E-unification est indécidable.

Toutefois, la satisfiabilité des systèmes de contraintes de l'intrus ne se réduit pas à l'unification en général comme nous l'avons montré sur l'exemple de la théorie vide : la satisfiabilité pour la théorie vide est NP-difficile bien que l'unification syntaxique soit dans la classe P.

Nous laissons ouverts les problèmes de l'existence d'une théorie E telle que

(1) l'E-unification soit décidable mais non la satisfiabilité, ou bien telle que

(2) la satisfiabilité soit décidable mais non la (S-)équivalence.

Concernant le premier point, un résultat approchant a été obtenu par S. Delaune [Del06a] pour la théorie des groupes abéliens avec homomorphisme (AGh). Toutefois, les problèmes de satisfiabilité considérés dans ce travail [Del06a] semblent strictement plus expressifs que les nôtres, car autorisant le filtrage des termes.

Travaux existants. L'outil Proverif [Blab, BAF05] fournit un semi-algorithme approché couvrant sensiblement les mêmes propriétés de sécurité que celles étudiées ici pour un nombre non borné de sessions. En cas d'échec de la tentative de preuve, l'outil permet de reconstruire une attaque [AB05b] avec de bonnes chances de succès en pratique. Toutefois, cette reconstruction ne couvre pas les propriétés d'équivalence.

À notre connaissance, en dehors de ces travaux, les algorithmes existants permettant de prouver des équivalences observationnelles entre processus cryptographiques concernent exclusivement le spi-calcul [Hüt00, DSV03, BN05], et par conséquent ne s'appliquent pas à un cadre équationnel. Ainsi, les résultats de ce chapitre constituent une première étude algorithmique, sans approximation, pour un nombre borné de sessions, des propriétés d'équivalence de processus en présence de théorie équationnelle arbitraire.

Perspectives. Les résultats obtenus ici permettent de réduire l'analyse d'un protocole fini à l'étude des problèmes correspondants (satisfiabilité ou S-équivalence) sur les systèmes de contraintes, ce pour une théorie équationnelle quelconque. Une question naturelle est donc la décidabilité de ces problèmes, et notamment de la S-équivalence, pour les théories équationnelles E « intéressantes » en cryptographie. À cet égard, nous consacrons le chapitre suivant à l'étude de la S-équivalence (positive) pour les théories équationnelles sous-terme-convergentes.

Parmi les différentes notions d'équivalence observationnelle disponibles, nous avons choisi de nous concentrer sur la diff-équivalence. L'expression de la sémantique symbolique correspondante nous a amené à introduire la notion de systèmes de contraintes de l'intrus, et le problème de la S-équivalence de ces systèmes. Plus généralement, on peut se poser la question de l'existence d'une réduction générique du même ordre allant de la may-équivalence ou de la congruence barbue vers la S-équivalence. Les travaux en cours de S. Delaune, S. Kremer et M. Ryan [Kre06] suggèrent des réponses positives à ces questions dans le cas du pi-calcul appliqué, pour la may-équivalence et une équivalence légèrement plus fine que la congruence barbue.

Chapitre 4

Résolution des systèmes de contraintes de l'intrus

Sommaire

L'objet de ce chapitre est l'étude de la satisfiabilité et de la S-équivalence positive des systèmes de contraintes de l'intrus, pour la classe des théories équationnelles sous-terme-convergentes.

Nous reprenons ici les notations du chapitre précédent en ce qui concerne les systèmes de contraintes de l'intrus (section 3.4). Rappelons qu'une théorie équationnelle est sous-terme-convergente (section 2.7) ssi elle est engendrée par un système de réécriture $\mathcal{R}$ convergent tel que pour toute règle $l \to r$ dans $\mathcal{R}$, r est ou bien un sous-terme strict de l, ou bien un terme clos en forme $\mathcal{R}$-normale.

Les sections qui suivent ont pour but de prouver le résultat suivant :

Théorème 4.1. *Soit E une théorie équationnelle engendrée par un système de réécriture sous-terme-convergent et fini. La satisfiabilité des systèmes de contraintes de l'intrus sur E est dans NP. La S-équivalence entre systèmes de contraintes de l'intrus positifs est dans la classe co-NP.*

Comme le veut l'usage, les classes de complexité s'entendent en fonction de la taille des systèmes de contraintes, le système de réécriture étant fixé.

Grâce au théorème 3.17, on en déduira que pour une telle théorie équationnelle, les propriétés d'accessibilité du chapitre précédent sont dans NP, de même que la (non-)diff-équivalence entre processus positifs (finis), et par conséquent l'existence d'attaques par dictionnaire sur les processus positifs (finis). Par ailleurs, nous savons déjà que tous ces problèmes sont NP-difficiles (section 3.5).

La preuve du théorème 4.1 repose sur une procédure de décision à base de transformations syntaxiques des systèmes de contraintes. Dans la section 4.1, nous présentons cette procédure et les différentes règles de transformation qui la composent. Nous prouvons ensuite dans les sections suivantes la correction, la complétude et la terminaison du système de règles (dans un sens précisé plus bas). Enfin, nous concluons la preuve à partir de l'analyse de la procédure présentée et du résultat de M. Abadi et et V. Cortier [AC04].

Convention. Dans la suite de ce chapitre, nous fixons une théorie équationnelle E engendrée par un système de réécriture convergent $\mathcal{R}$ — supposé fini en ce qui concerne les analyses de complexité. D'après la proposition 2.9, on peut supposer que les règles de $\mathcal{R}$ ne contiennent aucun symbole libre dans E, et notamment aucun paramètre $w_i^{\tau_i}$. Ainsi, les règles de $\mathcal{R}$ sont de la forme $l \to r$ où $l, r \in \mathcal{F}[\mathcal{X}^1]$ sont des termes du premier ordre.

Nous fixons également une certaine structure de système de contraintes de l'intrus, c'est-à-dire des paramètres $w_1^{\tau_1^*}, \ldots, w_{k_0}^{\tau_{k_0}^*}$ et un ensemble fini de variables du second ordre $\mathcal{Y} \subseteq \mathcal{X}$ tels que $\mathrm{maxar}(\mathcal{Y}) \leq k_0$.

Afin d'alléger les notations, dans la suite nous écrivons simplement w_i pour le i-ème paramètre fixé $w_i^{\tau_i^*}$. De plus, nous réservons les lettres M, $N\ldots$ aux termes du second ordre tels que $\mathrm{par}(M) \subseteq \{w_1, \ldots, w_{k_0}\}$ et $\mathrm{maxar}(M) \leq k_0$.

En outre, par souci de simplification, nous nous plaçons dans les hypothèses du point (4) de la proposition 3.12 :

> (*) Il existe une sorte non dégénérée τ_0^* telle que pour tout $\tau \in \mathcal{T} - \{\tau_0^*\}$ il existe un symbole libre public $h_\tau : \tau \to \tau_0^*$.

Ceci permet de ramener le problème de la S-équivalence de deux systèmes $\Sigma_1 = \Phi_1; \mathcal{C}_1$ et $\Sigma_2 = \Phi_2; \mathcal{C}_2$ à l'équivalence de Σ_1' et Σ_2' définis par :

$$\Sigma_i' \triangleq \Phi_i; \mathcal{C}_i \cup \{X^* \rhd^? x,\ Y^* \rhd^? y,\ x =_E^? y\} \qquad (i \in \{1,2\})$$

où $X^*, Y^*, x, y \in \mathrm{var}(\Sigma_i)$ sont des variables de sorte τ_0^*, et $\mathrm{ar}(X^*) = \mathrm{ar}(Y^*) = \mathrm{maxpar}(\Sigma_i) = k_0$.

Dans la suite, nous fixons deux variables X^*, Y^* de sorte τ_0^* et d'arité k_0, et nous appelons *standard* les systèmes de contraintes de l'intrus Σ ayant la forme de Σ'_i ci-dessus, c'est-à-dire contenant des contraintes $X^* \rhd^? x$, $Y^* \rhd^? y$, $x =^?_E y$ telles que les variables X^*, Y^*, x, y n'apparaissent nulle part ailleurs dans Σ.

La condition (*) ci-dessus ne constitue pas une réelle perte de généralité dans la mesure où, d'une part, $\mathcal{R}$ est convergent et par conséquent sans sortes dégénérées, et d'autre part, il est toujours possible d'étendre la signature par une sorte spéciale τ_0^* et des symboles libres publics $h_\tau : \tau \to \tau_0^*$ pour tout $\tau \neq \tau_0$. On pourra constater plus bas que les symboles h_τ n'interviennent pas au cours de la procédure mais uniquement dans la preuve.

4.1 Procédure de décision

Cette section a pour but de présenter la procédure de décision permettant d'établir le théorème 4.1. Nous commençons par introduire les *systèmes de contraintes étendus*. Fondamentalement, il s'agit de représenter syntaxiquement des ensembles (généralement infinis) de solutions d'un système de contraintes initial. Nous décrivons ensuite un ensemble de règles de transformation sur les systèmes étendus, dont nous montrons plus loin (section 4.3 et suivantes) la correction, la complétude et la terminaison lorsque $\mathcal{R}$ est sous-terme-convergent et fini.

4.1.1 Systèmes de contraintes étendus

Rappelons que, suivant les conventions précédentes (section 3.4 et préambule), les lettres t, $t_1 \ldots$ désignent des termes du premier ordre c'est-à-dire $t, t_1 \in \mathcal{F}[\mathcal{X}^1]$ tandis que M, N, $M_1 \ldots$ désignent des termes du second ordre dans $\mathcal{F}_{\text{pub}}[\mathcal{X}^2 \cup \{w_1, \ldots, w_{k_0}\}]$. Nous notons encore T, $T_1 \ldots$ pour des termes quelconques (bien sortés) de $\mathcal{F}[\mathcal{X}^1 \cup \mathcal{X}^2 \cup \mathcal{N} \cup \mathcal{W}]$, et V, $V_1 \ldots$ pour des variables quelconques de $\mathcal{X} = \mathcal{X}^1 \cup \mathcal{X}^2$. Les lettre C, D, $C_1 \ldots$ désignent comme auparavant des contextes publics, c'est-à-dire des termes de la forme $C \in \mathcal{F}_{\text{pub}}[w_1^{\tau_1}, \ldots, w_n^{\tau_n}]$ (en particulier clos). Nous utilisons les notations var(.), $\text{var}^1(.)$, $\text{var}^2(.)$, par(.), maxpar(.) et maxar(.) définies précédemment (section 3.4).

Dans ce chapitre, étant donnée une substitution σ, on note $\text{var}(\sigma) \triangleq \text{supp}(\sigma) \cup \text{var}(\text{supp}(\sigma)\,\sigma)$. Sauf mention du contraire, toutes les substitutions (partielles) σ, λ, $\mu \ldots$ considérées par la suite sont bien formées, c'est-à-dire qu'elles vérifient par convention :

- pour tout $x \in \text{dom}(\sigma)$, $x\sigma$ est un terme du premier ordre ;
- pour tout $X \in \text{dom}(\sigma)$, $X\sigma$ est un terme du second ordre tel que $\text{par}(X\sigma) \subseteq \{w_1, \ldots, w_{\text{ar}(X)}\}$.

Définition 4.1. Un *système de contraintes étendu*, ou simplement *système de contraintes* dans ce chapitre, est un quintuplet $\Sigma = \Phi ; \Psi ; \mathcal{C} ; \mathcal{N} ; \sigma$ où

- Φ est un ensemble fini d'expressions $M \rhd t$, appelées *règles d'environnement* (ou simplement *règles*) de Σ ;
- Ψ est un ensemble fini d'expressions $\forall \beta. M \bowtie N$ où $\beta \subseteq \mathcal{X}^2$, appelées *équations* de Σ ;
- $\mathcal{C}$ est un ensemble fini de contraintes de la forme $t_1 =^?_E t_2$, $t_1 \neq^?_E t_2$ et $X \rhd^? t$, où dans le dernier cas, la variable X apparaît une seule fois dans $\mathcal{C}$;

– $\mathcal{N}$ est un ensemble de termes du premier ordre t ;
– σ est une substitution idempotente de support disjoint de $\mathrm{var}(\Phi, \Psi, \mathcal{C}, \mathcal{N})$, autrement dit telle que $\Sigma\sigma = \Sigma$.

Toutes les règles, équations et contraintes sont supposées *bien sortées*, dans le sens où leurs membres gauche et droit sont de même sorte.

On pose $\mathrm{var}(\Sigma) = \mathrm{var}(\Phi, \Psi, \mathcal{C}, \mathcal{N}, \sigma) \cup \mathcal{Y}$. (Les variables de $\mathcal{Y}$ sont ajoutées par commodité en vue de la définition des solutions à venir.) Un système $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ est dit *positif* s'il ne contient pas de contraintes de différence $\neq_E^?$.

Les quantificateurs $\forall\beta$ dans les équations sont introduits ici pour des raisons techniques liées à la terminaison de la procédure. On applique les conventions habituelles sur les variables liées : les équations et plus généralement les termes quantifiés sont considérés modulo renommage (bien formé inversible) des variables liées et modulo suppression des quantificateurs inutiles : $(\forall\beta.M \bowtie N) = \forall(\beta \cap \mathrm{var}(M, N)).M \bowtie N$. Par commodité, on écrit $M \bowtie N$ plutôt que $\forall\emptyset.M \bowtie N$, et $\forall X_1, \ldots, X_n.M \bowtie N$ plutôt que $\forall\{X_1, \ldots, X_n\}.M \bowtie N$. De plus, on pose $\mathrm{var}(\forall\beta.M \bowtie N) = \mathrm{var}(M, N) - \beta$, et $(\forall\beta.M \bowtie N)\mu = (\forall\beta.M\mu \bowtie N\mu)$ si μ est une substitution telle que $\beta \cap \mathrm{var}(\mu) = \emptyset$.[1] Enfin, $\bowtie$ est vu comme un symbole commutatif.

Les composantes d'un système de contraintes $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ peuvent s'interpréter de la manière suivante. La composante Φ joue le même rôle que la composante du même nom dans les systèmes de contraintes de l'intrus (section 3.4) : chaque règle $M \rhd t$ de Φ correspond à un terme connu (ou *déductible* [AC04, AC06]) par l'attaquant via le terme du second ordre M, appelé *recette* ou *calcul* de t. Initialement, Φ est de la forme $\{w_1 \rhd t_1, \ldots, w_{k_0} \rhd t_{k_0}\}$. Durant la procédure, de nouveaux termes sont ajoutés. Par exemple, étant donnés $w_1 \rhd \mathsf{enc}(\mathsf{a}, \mathsf{b})$ et $w_2 \rhd \mathsf{b}$ dans Φ, et une règle $\mathsf{dec}(\mathsf{enc}(x, y), y) \to x$ dans $\mathcal{R}$, on pourra ajouter $\mathsf{dec}(w_1, w_2) \rhd \mathsf{a}$.

Les équations $\forall\beta.M \bowtie N$ de Ψ correspondent à des relations visibles par l'attaquant. La signification d'une telle équation est que pour toute substitution μ de support inclus dans β les deux calculs $M\mu$ et $N\mu$ donnent le même résultat (modulo E) lorsque que les paramètres w_i sont substitués par leurs valeurs dans Φ. Par exemple, étant données deux règles $w_1 \rhd \mathsf{h}(\mathsf{n})$ et $w_2 \rhd \mathsf{n}$ ($\mathsf{h} \in \mathcal{F}_{\mathrm{pub}}$), une équation visible est $w_1 \bowtie \mathsf{h}(w_2)$.

L'ensemble $\mathcal{C}$ est formé de contraintes identiques à celles des systèmes de contraintes de l'intrus du chapitre précédent. $\mathcal{N}$ représente un ensemble de contraintes de normalisation pour le système de réécriture $\mathcal{R}$. Enfin, la substitution σ est utilisée pour collecter les valeurs des *variables résolues* du problème.

Relations associées à $(\Phi, \mathcal{C})$ et à Ψ

Définition 4.2. Étant donnés un ensemble de règles d'environnement Φ et un ensemble de contraintes $\mathcal{C}$, on note $M \rhd_{\Phi,\mathcal{C}} t$ lorsqu'il existe un contexte public $C \in \mathcal{F}_{\mathrm{pub}}[w_1^{\tau_1}, \ldots, w_{m+n}^{\tau_{m+n}}]$ pour des types $\tau_1, \ldots, \tau_{m+n}$ arbitraires, des règles $M_1 \rhd t_1$, $\ldots$, $M_m \rhd t_m$ dans Φ et des contraintes $X_1 \rhd^? s_1$, $\ldots$, $X_n \rhd^? s_n$ tels que

$$\begin{aligned} M &= C[M_1, \ldots, M_m, X_1, \ldots, X_n] \\ t &= C[t_1, \ldots, t_m, s_1, \ldots, s_n] \end{aligned}$$

[1] De même, si μ est une substitution partielle de domaine fini contenant $\mathrm{var}(\forall\beta.M \bowtie N)$, on pose $(\forall\beta.M \bowtie N)\mu = (\forall\beta.M\mu' \bowtie N\mu')$ où $\mu' = \mu\|_{\mathrm{dom}(\mu) \uplus \beta}$ et $\mathrm{var}(\mathrm{im}(\mu)) \cap \beta = \emptyset$.

On écrit $M \rhd_\Phi t$ si $M \rhd_{\Phi,\emptyset} t$, et $M \rhd_\Phi^E t$ s'il existe t' tel que $M \rhd_\Phi t'$ et $t' =_E t$.

Notons que $M \rhd_{\Phi,\mathcal{C}} t$ implique $\mathrm{var}(M,t) \subseteq \mathrm{var}(\Phi,\mathcal{C})$, du fait que C est clos. De plus, lorsque Φ est de la forme $\Phi = \{w_1 \rhd t_1, \ldots, w_{k_0} \rhd t_{k_0}\}$, on a $M \rhd_\Phi t$ ssi $M[t_1,\ldots,t_{k_0}] = t$. Dans ce cas, on note également

$$M[\Phi] \triangleq M[t_1,\ldots,t_{k_0}].$$

Étant donné un ensemble d'équations Ψ, on définit la congruence $\bowtie_\Psi$ de la manière suivante. Étant donnés deux termes du second ordre M_1 et M_2, notons $M_1 \leftrightarrow_\Psi M_2$ lorsqu'il existe une équation $\forall\beta.M \bowtie N$ dans Ψ, une position p et une substitution bien formée μ de support inclus dans β telle que $M_1|_p = M\mu$ et $M_2 = M_1[N\mu]_p$. Par définition, $\bowtie_\Psi$ est la clôture réflexive et transitive de $\leftrightarrow_\Psi$.

Notons que lorsque Ψ est clos, c'est-à-dire $\mathrm{var}(\Psi) = \emptyset$, la congruence $\bowtie_\Psi$ est stable par substitution, et induit donc une théorie équationnelle. Plus précisément, on dispose de la propriété suivante :

Lemme 4.2. *Soit Ψ un ensemble d'équations $\forall\beta.M \bowtie N$. Soit M_1, M_2 deux termes du second ordre, et λ une substitution. Alors,*

$$M_1 \bowtie_\Psi M_2 \quad\Rightarrow\quad M_1\lambda \bowtie_{\Psi\lambda} M_2\lambda$$

Démonstration. On procède par induction sur la longueur de la chaîne $M_1 \leftrightarrow_\Psi^* M_2$.

Si $M_1 \leftrightarrow_\Psi M_2$ c'est-à-dire $M_1|_p = M\mu$ et $M_2 = M_1[N\mu]_p$ pour une certaine position p, règle $(\forall\beta.M \bowtie N) \in \Psi$ et substitution μ de support inclus dans β, alors quitte à renommer les variables de β on peut supposer $\beta \cap \mathrm{var}(\lambda) = \emptyset$, d'où : $(M_1\lambda)|_p = M\mu\lambda = M\lambda\mu'$, $M_2\lambda = (M_1\lambda)[N\lambda\mu']_p$ et $(\forall\beta.M \bowtie N)\lambda = \forall\beta.M\lambda \bowtie N\lambda$ où $\mu' = \{X \mapsto (X\mu)\lambda\}_{X\in\mathrm{supp}(\mu)}$. Par conséquent $M_1\lambda \bowtie_{\Psi\lambda} M_2\lambda$. □

Enfin, le lemme suivant permet de transposer certaines équivalences modulo Ψ en égalités modulo E.

Lemme 4.3. *Soit $\Phi = \{w_1 \rhd t_1, \ldots, w_{k_0} \rhd t_{k_0}\}$ et Ψ un ensemble d'équations tels que pour tout $\forall\beta.M \bowtie N$ dans Ψ, $M[\Phi] =_E N[\Phi]$. Alors pour tous termes M_1, M_2 (vérifiant par convention $\mathrm{par}(M_1,M_2) \subseteq \{w_1,\ldots,w_{k_0}\}$), on a*

$$M_1 \bowtie_\Psi M_2 \;\Rightarrow\; M_1[\Phi] =_E M_2[\Phi].$$

Démonstration. Comme précédemment, on procède par induction sur la longueur de la chaîne

$$M_1 \leftrightarrow_\Psi^* M_2.$$

Si $M_1 \leftrightarrow_\Psi M_2$, c'est-à-dire $M_1|_p = M\mu$ et $M_2 = M_1[N\mu]_p$ pour une certaine position p, une règle $(\forall\beta.M \bowtie N) \in \Psi$ et une substitution μ de support inclus dans β, alors l'hypothèse $M[\Phi] =_E N[\Phi]$ implique

$$(M\mu)[\Phi] =_E (N\mu)[\Phi]$$

par application de la substitution $\{X \mapsto (X\mu)[\Phi]\}_{X\in\mathrm{supp}(\mu)}$. On en déduit la propriété escomptée : $M_1[\Phi] =_E M_2[\Phi]$. □

Solutions d'un système de contraintes étendu

Nous donnons maintenant une sémantique formelle aux systèmes de contraintes à l'aide de la notion suivante de solution.

Définition 4.3. Une substitution partielle close (bien formée) θ de domaine $\mathcal{Y}$ est une *solution* de $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$, ce que l'on écrit $\theta \models \Sigma$, ss'il existe une substitution partielle close (bien formée) λ de domaine $\mathrm{var}(\Sigma) = \mathrm{var}(\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma) \cup \mathcal{Y}$ telle que :

(i) pour chaque contrainte $X \rhd^? t$ dans $\mathcal{C}$, on a $X\lambda \rhd_{\Phi\lambda} t\lambda$;

(ii) pour chaque $t_1 =^?_E t_2$ dans $\mathcal{C}$, $t_1\lambda =_E t_2\lambda$;

(iii) pour chaque $t_1 \neq^?_E t_2$ dans $\mathcal{C}$, $t_1\lambda \neq_E t_2\lambda$;

(iv) pour chaque $t \in \mathcal{N}$, $t\lambda$ est en forme $\mathcal{R}$-normale ;

(v) λ *étend* σ, au sens où pour tout $V \in \mathrm{dom}(\lambda)$, $V\sigma\lambda = V\lambda$;

(vi) λ est *liée* à θ, au sens où pour tout $X \in \mathcal{Y}$, on a $X\theta \bowtie_{\Psi\lambda} X\lambda$.

On écrit $\theta, \lambda \models \Sigma$ pour spécifier une substitution partielle λ liée à la solution θ.

Le premier point (i) signifie, intuitivement, que pour tout $X \rhd^? t$ dans $\mathcal{C}$, $X\lambda$ doit être un calcul de $t\lambda$ via les règles de $\Phi\lambda$. Ces contraintes ne sont pas considérées modulo E, c'est pourquoi l'ensemble Φ devra être complété par de nouvelles règles au fur et à mesure l'algorithme. Les points (ii) et (iii) requiert la satisfaction des contraintes d'égalité et de différence modulo E au sens usuel. Le point (iv) interprète les contraintes de normalisation de $\mathcal{N}$. Le point (v) fixe la valeur de λ sur les variables résolues du système, déterminées par σ. Enfin, le point (vi) relie θ à $\lambda|_{\mathcal{Y}}$ via les équations de $\Psi\lambda$. Au cours de la preuve de complétude, avec l'ajout de nouvelles équations dans Ψ, ceci nous permet de modifier $\lambda|_{\mathcal{Y}}$ sans « perdre » la solution θ donnée au départ.

Notons que les valeurs de σ prises sur les variables de $\mathrm{supp}(\sigma) - \mathcal{Y}$ ne jouent pas de rôle dans la définition des solutions θ, car ces valeurs ne sont pas mises en jeu dans les points (i), (ii), (iii), (iv) et (vi) du fait de $\mathrm{supp}(\sigma) \cap \mathrm{var}(\Phi; \Psi; \mathcal{C}; \mathcal{N}) = \emptyset$. Toutefois, nous conservons les valeurs de ces variables dans les systèmes de contraintes afin de faciliter les raisonnements sur les règles de transformation.

Convention. Un système de contraintes de l'intrus $\Sigma = \Phi;\ \mathcal{C}$ (définition 3.12) est vu comme un système de contraintes étendu $\Sigma = \Phi;\ \emptyset;\ \mathcal{C};\ \emptyset;\ \{\}$ où $\{\}$ désigne la substitution vide (*i.e.* la fonction identité sur $\mathcal{X}$).

Rappelons que par définition, dans ce cas, Φ et $\mathcal{C}$ sont de la forme

$$\Phi = \{w_1 \rhd t_1, \ldots, w_{k_0} \rhd t_{k_0}\} \quad \text{et}$$
$$\mathcal{C} = \{X_1 \rhd^? x_1, \ldots, X_m \rhd^? x_m,\ s_1 \sim_1 s'_1, \ldots, s_n \sim_n s'_n\}$$

où $\{X_1, \ldots, X_m\} = \mathcal{Y}$ est l'ensemble fixé précédemment, $\sim_i \in \{=^?_E, \neq^?_E\}$ et où les *conditions de régularité* suivantes sont vérifiées :

(1) les variables $X_1, \ldots, X_m$ et $x_1, \ldots, x_m$ sont deux à deux disjointes, et $\mathrm{var}(\Sigma) = \{X_1, \ldots, X_m, x_1, \ldots, x_m\}$;

(2) pour tous $1 \leq k \leq k_0$ et $x \in \mathrm{var}(t_k)$, il existe une contrainte $(X_i \rhd^? x_i) \in \mathcal{C}$ telle que $\mathrm{ar}(X_i) < k$.

De plus, en supposant sans perte de généralité que $0 \leq \mathrm{ar}(X_1) \leq \ldots \leq \mathrm{ar}(X_m) \leq k_0$, la condition (2) équivaut à pour tout $1 \leq i \leq m$,

$$\mathrm{var}(t_1, \ldots, t_{\mathrm{ar}(X_i)}) \subseteq \{x_1, \ldots, x_{i-1}\}.$$

Cette convention est justifiée par le fait suivant.

Fait 4.4. *Les ensembles de solutions d'un système de contraintes de l'intrus Σ selon les deux définitions 3.12 et 4.3 coïncident.*

De plus, pour toute solution θ de Σ, il existe un unique λ tel que $\theta, \lambda \models \Sigma$.

Démonstration. Soit θ une solution de Σ vu comme système de contraintes de l'intrus, avec les notations ci-dessus, où l'on suppose $0 \leq \mathrm{ar}(X_1) \leq \ldots \leq \mathrm{ar}(X_m) \leq k_0$. Il existe une solution au premier ordre λ^1 associée à θ :

- pour tout $1 \leq i \leq m$, $(X_i\,\theta)[t_1\lambda^1, \ldots, t_{\mathrm{ar}(X_i)}\lambda^1] = x_i\,\lambda^1$, et
- pour tout $1 \leq j \leq n$, $s_j\,\lambda^1 =_E s'_j\,\lambda^1$ ssi $(\sim_j) = (=_E^?)$.

Soit $\lambda = \theta \uplus \lambda^1$. On a bien $\theta, \lambda \models \Sigma$ où Σ est vu comme un système étendu.

Réciproquement, soit θ et λ tels que $\theta, \lambda \models \Sigma$.

Pour tout $1 \leq i \leq m$, on a $X_i\lambda \rhd_{\Phi\lambda} x_i\lambda$, autrement dit

$$x_i\lambda = (X_i\lambda)[t_1\lambda, \ldots, t_{\mathrm{ar}(X_i)}\lambda].$$

Or, d'après la dernière condition, pour tout i, $X_i\lambda = X_i\theta$. Par conséquent, θ est bien solution de Σ en tant que système de contraintes de l'intrus.

Enfin, étant données les conditions de régularité vérifiées par Σ, l'unicité de λ découle par induction sur i des équations $x_i\,\lambda = (X_i\,\theta)[t_1\lambda, \ldots, t_{\mathrm{ar}(X_i)}\lambda]$ et $X_i\lambda = X_i\theta$. □

Système de contraintes particuliers

Un système de contraintes correspondant à un système de contraintes de l'intrus est dit *initial*. Outre les systèmes initiaux, on distingue les systèmes *pré-résolus*, *résolus* et *standard* au sens suivant.

Définition 4.4. Un système de contraintes $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ est *pré-résolu* ssi $\mathcal{C}$ est de la forme ci-dessus, c'est-à-dire ssi les membres droits de contraintes de déductibilité $(X \rhd^? t) \in \mathcal{C}$ sont des variables deux à deux disjointes.

Un tel système Σ est *résolu* ss'il est pré-résolu et si

(i) $\mathcal{C}$ ne contient pas de contraintes d'égalité $s =_E^? s'$,

(ii) pour toute contrainte de différence $s \neq_E^? s'$ dans $\mathcal{C}$, on a $s \neq_E s'$, et

(iii) pour tout terme $t \in \mathcal{N}$, t est en forme $\mathcal{R}$-normale.

Enfin, un système $\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ est *standard* (toujours pour X^* et Y^* fixés d'arité k_0) ssi l'une des conditions suivantes est vérifiées :

(a) $\mathcal{C}$ est de la forme $\mathcal{C} = \mathcal{C}_0 \uplus \{X^* \rhd^? x, Y^* \rhd^? y, x =_E^? y\}$ où $X^*, Y^*, x, y \notin \mathrm{var}(\Phi;\ \Psi;\ \mathcal{C}_0;\ \mathcal{N};\ \sigma)$;

(b) $\mathcal{C}$ et σ sont de la forme $\mathcal{C} = \mathcal{C}_0 \uplus \{X^* \rhd^? x, Y^* \rhd^? x\}$ et $\sigma = \sigma_0\{y \mapsto x\}$ où $X^*, Y^*, x, y \notin \mathrm{var}(\Phi;\ \Psi;\ \mathcal{C}_0;\ \mathcal{N};\ \sigma_0)$;

(c) $\mathcal{C}$ et σ sont de la forme $\mathcal{C} = \mathcal{C}_0 \uplus \{X^* \rhd^? x\}$, et $\sigma = \sigma_0\{y \mapsto x, Y^* \mapsto X^*\}$ où $X^*, Y^*, x, y \notin \mathrm{var}(\Phi;\ \Psi;\ \mathcal{C}_0;\ \mathcal{N};\ \sigma_0)$, quitte à inverser les rôles de X^* et de Y^*.

Les conditions techniques de la dernière définition tiennent compte des transformations possibles d'un système initial standard au cours de la procédure.

L'intérêt de ces diverses notions réside notamment dans la proposition suivante, qui nous permet d'établir plus bas la structure générale de la procédure de décision.

Proposition 4.5. *On a les faits suivants :*

1. *Tout système de contraintes résolu est satisfiable, i.e. admet au moins une solution.*
2. *Étant donné un oracle pour le problème du mot dans E, on peut décider en temps polynomial l'inclusion des solutions d'un système résolu et standard Σ dans celles d'un système initial, standard et positif Σ'.*

Démonstration. **1.** Dans la mesure où la signature contient un nombre infini de constantes libres publiques de chaque sorte, on construit une solution à tout système résolu $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ de la manière suivante.

Soit λ^1 une substitution partielle close associant à chaque variable du premier ordre $x \in \mathrm{var}(\Sigma\sigma) = \mathrm{var}(\Phi, \mathcal{C}, \Psi, \mathcal{N}, \mathrm{supp}(\sigma)\sigma, \mathcal{Y}\sigma)$ une constante publique fraîche c_x de sorte appropriée. Pour toute variable $X \in \mathrm{var}(\Sigma\sigma)$, on pose $X\lambda^2 = x\lambda^1 = c_x$ s'il existe une contrainte $X \rhd^? x$ dans $\mathcal{C}$ (une telle contrainte est unique par définition des systèmes de contraintes). Dans le cas contraire, on pose $X\lambda^2 = c_X$, où c_X est une constante publique libre fraîche de même sorte que X. Dans le second cas, notons que X n'apparaît pas dans $\mathcal{C}$ du fait que Σ est pré-résolu.

Soit $\lambda_0 = \sigma|_{\mathrm{var}(\Sigma)}\,(\lambda^1 \uplus \lambda^2)$ et $\theta_0 = \lambda_0|_{\mathcal{Y}}$. Le système Σ étant résolu, vérifions que $\theta_0, \lambda_0 \models \Sigma$. En effet,

- pour tout $X \rhd^? t$ dans $\mathcal{C}$, $t = x \in \mathcal{X}^1$ donc $X\lambda_0 = c_x \rhd_{\Phi\lambda_0} c_x = t_{\lambda_0}$;
- $\mathcal{C}$ ne contient pas de contraintes $t_1 =^?_E t_2$;
- pour tout $t_1 \neq^?_E t_2$ dans $\mathcal{C}$, l'hypothèse $t_1 \neq_E t_2$ implique $t_1\lambda_0 \neq_E t_2\lambda_0$ par stabilité de E par remplacement de constantes libres (proposition 2.5) ;
- pour chaque $t \in \mathcal{N}$, comme t est en forme $\mathcal{R}$-normale, $t\lambda_0$ l'est également car par hypothèse les règles de $\mathcal{R}$ ne comportent pas de symbole libre ;
- par construction $\sigma|_{\mathrm{dom}(\lambda_0)}\lambda_0 = \lambda_0$ et $\theta \bowtie_{\Psi\lambda_0} \lambda_0|_{\mathcal{Y}}$.

2. Soit $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ un système résolu et standard, et $\Sigma' = \Phi';\ \emptyset;\ \mathcal{C}';\ \emptyset,\ \sigma'$ un système initial, standard et positif. Posons $\Phi' = \{w_1 \rhd t_1, \ldots, w_{k_0} \rhd t_{k_0}\}$. Nous montrons l'équivalence des deux conditions suivantes :

(1) Il existe θ telle que $\theta \models \Sigma$ et $\theta \not\models \Sigma'$.

(2) Soit θ_0 et λ_0 construits à partir du système résolu Σ comme dans le point 1. ci-dessus. Si $\theta_0 \models \Sigma'$ alors il existe une équation $\forall\beta.M \bowtie N$ dans Ψ telle que

$$M[\Phi'\lambda'_0] \neq_E N[\Phi'\lambda'_0]$$

où λ'_0 est l'unique substitution partielle telle que $\theta_0, \lambda'_0 \models \Sigma'$.

Cette équivalence implique bien la propriété voulue dans la mesure où le point (2) est décidable en temps polynomial dans la taille des systèmes Σ et Σ' avec oracle pour $=_E$. En effet, θ_0, λ_0 et λ'_0 sont constructibles en temps polynomial (voir le fait 4.4) et de tailles polynomiales. Enfin, l'oracle pour $=_E$ permet de vérifier en temps polynomial les contraintes de $\theta, \lambda'_0 \models \Sigma'$ ainsi que la validité de chaque équation de Ψ, interprétée modulo E dans $\Phi'\lambda'_0$.

Non (2) ⇒ non (1). Soit θ_0 et λ_0 construits à partir de Σ comme dans le point 1. Par hypothèse, il existe λ'_0 tel que $\theta_0, \lambda'_0 \models \Sigma'$ et que pour toute équation $\forall\beta.M \bowtie N$ dans Ψ, on a

$$M[\Phi'\lambda'_0] =_E N[\Phi'\lambda'_0] \tag{4.1.1}$$

Soit θ une solution de Σ pour un certain λ tel que $\theta, \lambda \models \Sigma$. Montrons que θ est également solution de Σ' pour un certain λ'. Quitte à renommer les constantes libres utilisées pour définir θ_0 et λ_0, dans la suite on suppose que ces constantes n'apparaissent pas dans l'image de θ et λ.

Posons $\mathcal{Y} = \{X_1, \ldots, X_m, X^*, Y^*\}$ avec

$$0 \leq \mathrm{ar}(X_1) \leq \ldots \leq \mathrm{ar}(X_m) \leq \mathrm{ar}(X^*) = \mathrm{ar}(Y^*) = k_0.$$

Sachant que Σ' est initial, standard et positif, $\mathcal{C}'$ est de la forme

$$\mathcal{C}' = \{X_1 \rhd^? x_1, \ldots, X_m \rhd^? x_m, X^* \rhd^? x, Y^* \rhd^? y \\ s_1 =^?_E s'_1, \ldots, s_n =^?_E s'_n, x =^?_E y\}.$$

où les conditions de régularité s'écrivent :

(1) les variables x_1, …, x_m, x, y sont deux à deux disjointes et vérifient $\mathrm{var}(s_1, s'_1, \ldots, s_n, s'_n, t_1, \ldots, t_{k_0}) \subseteq \{x_1, \ldots, x_m\}$;

(2) pour tous $1 \leq k \leq k_0$ et $x \in \mathrm{var}(t_k)$, il existe une contrainte $(X_i \rhd^? x_i) \in \mathcal{C}$ telle que $\mathrm{ar}(X_i) < k$.

Soit λ' de domaine $\mathrm{var}(\Sigma')$ défini par $\lambda'|_{\mathcal{Y}} = \theta$ et par induction, d'après la condition de régularité (2), pour tout $1 \leq i \leq m$,

$$x_i\lambda' = (X_i\theta)[t_1\lambda', \ldots, t_{\mathrm{ar}(X_i)}\lambda']$$

Soit $\rho^2 = \{c_x \mapsto X\}_{(X \rhd^? x) \in \mathcal{C}} \uplus \{c_X \mapsto X\}_{X \in \mathrm{var}(\Sigma) - \mathrm{var}(\mathcal{C})}$ le remplacement inverse de λ^2 défini dans le point 1. ci-dessus, et

$$\rho^1 = \{\, c \mapsto (c\rho^2\lambda)[\Phi'\lambda'_0] \,\}_{c \in \mathrm{dom}(\rho^2)}$$

Montrons par induction sur i que $x_i\lambda' =_E x_i\lambda'_0\rho^1$. Par stabilité de E par remplacement de symboles libres (proposition 2.5), sachant que $\theta_0, \lambda'_0 \models \Sigma'$ et vu la forme de Σ' (initial, positif), on en déduit en effet immédiatement que $\theta, \lambda' \models \Sigma'$.

De la définition $x_i\lambda' = (X_i\theta)[t_1\lambda', \ldots, t_{\mathrm{ar}(X_i)}\lambda']$, sachant que θ est bien formée et que $\mathrm{var}(t_1, \ldots, t_{\mathrm{ar}(X_i)}) \subseteq \{x_1, \ldots, x_{i-1}\}$, on déduit par hypothèse d'induction :

$$\begin{aligned} x_i\lambda' &= (X_i\theta)[t_1\lambda'_0\rho^1, \ldots, t_{\mathrm{ar}(X_i)}\lambda'_0\rho^1] \\ &= (X_i\theta)[\Phi'\lambda'_0\,\rho^1] \\ &= (X_i\theta)[\Phi'\lambda'_0]\,\rho^1 \end{aligned}$$

car $X_i\theta$ ne contient pas de constantes libres du domaine de ρ^1. D'après le lemme 4.3, l'hypothèse sur les équations de Ψ (équation 4.1.1) et le fait que $\theta, \lambda \models \Sigma$, ou plus précisément $X_i\lambda \bowtie_{\Psi\lambda} X_i\theta$, impliquent

$$(X_i\theta)[\Phi'\lambda'_0] =_E (X_i\lambda)[\Phi'\lambda'_0],$$

d'où :

$$x_i\lambda' =_E (X_i\lambda)[\Phi'\lambda'_0]\,\rho^1$$

Or, du fait de $\lambda_0|_{\mathcal{Y}} = \sigma|_{\mathcal{Y}}\lambda^2$ et $\lambda^2\rho^2 = \{\}_{\mathrm{dom}(\lambda^2)}$, on a $\sigma|_{\mathcal{Y}} = \lambda_0|_{\mathcal{Y}}\rho^2 = \theta_0|_{\mathcal{Y}}\rho^2$. Sachant que $\sigma|_{\mathcal{Y}}\lambda = \lambda$, on obtient

$$x_i\lambda' =_E (X_i\theta_0\rho^2\lambda)[\Phi'\lambda'_0]\,\rho^1$$

puis, par définition de ρ^1,

$$\begin{array}{rcl} x_i\lambda' & =_E & (X_i\theta_0\rho^1)[\Phi'\lambda'_0]\,\rho^1 \\ & = & (X_i\lambda_0)[\Phi'\lambda'_0]\,\rho^1 \end{array}$$

Enfin, comme $\theta_0, \lambda'_0 \models \Sigma'$ et que Σ' est initial, on a $\lambda'_0|_{\mathcal{Y}} = \theta_0$ et

$$x_i\lambda'_0 = (X_i\theta_0)[\Phi'\lambda'_0],$$

d'où finalement $x_i\lambda' =_E x_i\lambda'_0\rho^1$.

(2) ⇒ (1). Soit θ_0 et λ_0 construits comme précédemment à partir du système résolu Σ. On a en particulier $\theta_0, \lambda_0 \models \Sigma$.

Si $\theta_0 \not\models \Sigma'$ alors le point (1) est clair. Dans le cas contraire, $\theta_0, \lambda'_0 \models \Sigma'$ pour un certain λ'_0, et par hypothèse il existe une équation $\forall\beta.M \bowtie N$ dans Ψ telle que

$$M[\Phi'\lambda'_0] \neq_E N[\Phi'\lambda'_0]$$

À partir de θ_0, on construit une solution θ_1 de Σ telle $\theta_1 \not\models \Sigma'$ de la manière suivante. Reprenons les notations du point précédent concernant la forme de Σ'. Par hypothèse, $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ est standard et résolu. Ceci implique que $\mathcal{C}$ et σ sont de la forme (c) :

$\mathcal{C} = \mathcal{C}_0 \uplus \{X^* \rhd^? x\}$ et $\sigma = \sigma_0\{y \mapsto x,\ Y^* \mapsto X^*\}$ où $X^*, Y^*, x, y \notin \mathrm{var}(\Phi;\ \Psi;\ \mathcal{C}_0;\ \mathcal{N};\ \sigma_0)$ (quitte à inverser les rôles de X^* et Y^*).

Soit θ_1 défini par

$$\theta_1 = \theta_0|_{\mathrm{var}(\Sigma)-\{X^*,Y^*\}} \uplus \{X^* \mapsto M\mu\theta_0,\ Y^* \mapsto N\mu\theta_0\}$$

où μ est une substitution partielle close de domaine β envoyant chaque $X \in \beta$, sur une constante publique libre fraîche de même sorte que X. Comme $\forall\beta.M \bowtie N$ est dans Ψ et par construction de μ, on a $M\mu\theta_0 \bowtie_{\Psi\theta_0} N\mu\theta_0$.

Posons

$$\lambda_1 = \lambda_0|_{\mathrm{var}(\Sigma)-\{X^*,Y^*\}} \uplus \{X^* \mapsto M\mu\theta_0,\ Y^* \mapsto M\mu\theta_0\}$$

Sachant que $X^*, Y^* \notin \mathrm{var}(\Psi)$, on a $\Psi\lambda_1 = \Psi\lambda_0$. Par conséquent, pour tout $X \in \mathcal{Y} - \{X^*, Y^*\}$, on a

$$X\theta_1 = X\theta_0 \bowtie_{\Psi\lambda_1} X\lambda_0 = X\lambda_1$$

et de plus, pour tout $X \in \{X^*, Y^*\}$, $X\theta_1 \bowtie_{\Psi\lambda_1} X\lambda_1$. D'après la forme de Σ, $\theta_0, \lambda_0 \models \Sigma$ implique donc $\theta_1, \lambda_1 \models \Sigma$.

Pour finir vérifions que $\theta_1 \not\models \Sigma'$. En effet, par hypothèse $M[\Phi'\lambda'_0] \neq_E N[\Phi'\lambda'_0]$. Or, par définition, $\theta_0 = \lambda_0|_{\mathcal{Y}} = \sigma|_{\mathcal{Y}}\lambda^2$ où λ^2 envoie les variables non résolues X du système Σ sur des constantes publiques fraîches. Comme $\mathrm{supp}(\sigma) \cap \mathrm{var}(\Sigma) = \emptyset$,

on a $M\theta_0 = M\sigma\lambda^2 = M\lambda^2$. Par stabilité de E par remplacement des symboles libres (proposition 2.5) et par construction de μ, on en déduit $(M\mu\theta_0)[\Phi'\lambda'_0] \neq_E (N\mu\theta_0)[\Phi'\lambda'_0]$, autrement dit

$$(X^*\theta_1)[\Phi'\lambda'_0] \neq_E (Y^*\theta_1)[\Phi'\lambda'_0].$$

Par ailleurs, Σ' étant initial et standard, il est de la forme (a) :

$$\mathcal{C}' = \mathcal{C}'_0 \uplus \{X^* \rhd^? x',\ Y^* \rhd^? y',\ x' =^?_E y'\}$$

où $X^*, Y^*, x', y' \notin \mathrm{var}(\Phi;\ \Psi;\ \mathcal{C}_0;\ \mathcal{N};\ \sigma)$.

Supposons par contradiction que $\theta_1, \lambda'_1 \models \Sigma'$. Du fait que X^* et Y^* sont d'arité maximale k_0, les valeurs de λ'_1 sur $\mathrm{var}^1(\Sigma') - \{x', y'\}$ sont déterminées par la solution partielle $\theta_1|_{\mathcal{Y}-\{X^*,Y^*\}} = \theta_0|_{\mathcal{Y}-\{X^*,Y^*\}}$ (voir la preuve du fait 4.4). Ainsi,

$$\lambda'_0|_{\mathrm{var}^1(\Sigma')-\{x',y'\}} = \lambda'_1|_{\mathrm{var}^1(\Sigma')-\{x',y'\}}$$

et par conséquent, $\Phi'\lambda'_1 = \Phi'\lambda'_0$ du fait de $x', y' \notin \mathrm{var}(\Phi')$. On en déduit

$$x'\lambda'_1 = (X^*\theta_1)[\Phi'\lambda'_1] = (X^*\theta_1)[\Phi'\lambda'_0] \neq_E (Y^*\theta_1)[\Phi'\lambda'_0] = (Y^*\theta_1)[\Phi'\lambda'_1] = y'\lambda'_1$$

Ainsi, la contrainte $x' =^?_E y'$ n'est pas satisfaite par λ'_1 et $\theta_1 \not\models \Sigma'$. □

4.1.2 Principe de la procédure

La proposition précédente 4.5 nous incite à rechercher un ensemble de règles de transformations sur les systèmes de contraintes étendus qui soit correct et complet dans les sens suivants. Notons $\Longrightarrow$ la relation correspondante. Pour tout système initial Σ_0, on souhaite que

- (*correction*) pour tout Σ, si $\Sigma_0 \Longrightarrow^* \Sigma$ et $\theta \models \Sigma$ alors $\theta \models \Sigma_0$, et que
- (*complétude*) pour tout $\theta \models \Sigma_0$, il existe un système résolu Σ tel que $\Sigma_0 \Longrightarrow^* \Sigma$ et $\theta \models \Sigma$.

En effet, supposons en outre que

- la relation $\Longrightarrow$ préserve les systèmes standard (*i.e.* Σ standard et $\Sigma \Longrightarrow \Sigma'$ implique Σ' standard),
- qu'elle est effective (*i.e.* il existe un algorithme énumérant les successeurs d'un système par $\Longrightarrow$), et
- que l'on dispose d'une procédure de décision pour le problème du mot dans E.

Alors un tel système de règles procure un semi-algorithme pour tester la non-inclusion des solutions de deux systèmes initiaux standard Σ_0 et Σ_1 (Σ_1 positif) : il suffit d'énumérer les systèmes de contraintes résolus Σ accessibles depuis Σ_0, et d'appliquer la procédure de la proposition 4.5 à chaque Σ et Σ_1.

Enfin, si la relation $\Longrightarrow$ est à branchement fini et termine, alors par le lemme de König, le nombre de systèmes de contraintes atteignables étant fini, la procédure ci-dessus termine toujours.

D'après les remarques préliminaires de ce chapitre, on en déduit une procédure pour tester la S-équivalence de deux systèmes de contraintes de l'intrus positifs Σ_1 et Σ_2 : il suffit de tester la double inclusion des solutions des systèmes positifs standard Σ'_1 et Σ'_2, obtenus comme précédemment en ajoutant des contraintes $X^* \rhd^? x$, $Y^* \rhd^? y$, $x =_E y$ à Σ_1 et Σ_2.

4.1.3 Règles de transformation des systèmes de contraintes

Nous décrivons maintenant une procédure de transformation des systèmes de contraintes dont on montrera plus loin qu'elle est correcte pour tout système de réécriture $\mathcal{R}$, est complète pour tout système $\mathcal{R}$ sous-terme-convergent, et termine pour tout système $\mathcal{R}$ sous-terme-convergent *et* fini. Après quelques définitions, nous présentons la procédure proprement dite, donnée sous la forme d'un ensemble de règles de transformation et d'une stratégie d'application de ces règles.

Décompositions d'une règle de réécriture

Définition 4.5. Soit $\mathcal{R}$ un système de réécriture et $l \to r$ une règle de $\mathcal{R}$. On appelle *décomposition* de $l \to r$ la donnée d'un contexte public D de paramètres $\mathrm{par}(D) = \{w_1^{\tau_1}, \ldots, w_{n+p+q}^{\tau_{n+p+q}}\}$ pour certains types $\tau_1, \ldots, \tau_{n+p+q}$, et vérifiant

$$l = D[l_1, \ldots, l_n, y_1, \ldots y_p, z_1, \ldots, z_q]$$

pour $n \geq 0$ termes distincts non variables $l_1, \ldots, l_n$, $p \geq 0$ variables distinctes $y_1, \ldots, y_p \in \mathrm{var}(l_1, \ldots, l_n)$, et $q \geq 0$ variables distinctes $z_1, \ldots, z_q \notin \mathrm{var}(l_1, \ldots, l_n)$.

Une décomposition D est *propre* ssi ce n'est pas un paramètre, autrement dit $D \neq w_1^{\tau_1}$.

Notons que les termes $l_1, \ldots, l_n$ et les variables $y_1, \ldots, y_p, z_1, \ldots, z_q$ sont décrits de manière univoque par l et D. Dans la suite, nous identifions deux décompositions D et D' égales modulo renommage (*i.e.* remplacement inversible) des paramètres.

Dans les exemples qui suivent, tous les symboles sont supposés publics et pris sur une signature possédant une seule sorte τ_0^*.

Exemple 4.1. Considérons la règle de réécriture $\mathsf{dec}(\mathsf{enc}(x, y), y) \to x$ représentant le chiffrement symétrique déterministe. Cette règle possède deux décompositions propres :

- $D_1 = \mathsf{dec}(\mathsf{enc}(w_1, w_2), w_2)$ pour $n = p = 0$, $q = 2$, $z_1 = x$, $z_2 = y$;
- $D_2 = \mathsf{dec}(w_1, w_2)$ pour $n = p = 1$, $q = 0$, $l_1 = \mathsf{enc}(x, y)$, $y_1 = y$.

Exemple 4.2. Considérons la règle de réécriture $\mathsf{pdec}(\mathsf{penc}(x, \mathsf{pub}(y), z), y) \to x$ modélisant le chiffrement asymétrique probabiliste. Cette règle possède trois décompositions propres :

- $D_1 = \mathsf{pdec}(\mathsf{penc}(w_1, \mathsf{pub}(w_2), w_3), w_2)$ pour $n = p = 0$, $q = 3$, $z_1 = x$, $z_2 = y$, $z_3 = z$;
- $D_2 = \mathsf{pdec}(\mathsf{penc}(w_3, w_1, w_4), w_2)$ pour $n = 1$, $p = 1$, $q = 3$, $l_1 = \mathsf{pub}(y)$, $y_1 = y$, $z_1 = x$, $z_2 = z$;
- $D_3 = \mathsf{pdec}(w_1, w_2)$ pour $n = 1$, $p = 1$, $q = 0$, $l_1 = \mathsf{penc}(x, \mathsf{pub}(y), z)$, $y_1 = y$.

La notion de décomposition se justifie notamment par la propriété suivante.

Lemme 4.6. *Soit C un contexte public et $l \to r$ une règle de $\mathcal{R}$. Supposons qu'il existe des termes $t_1, \ldots, t_m$ et une substitution σ tels que*

$$C[t_1, \ldots, t_m] = l\sigma.$$

Alors il existe

- *une décomposition (avec les notations précédentes)*

$$l = D[l_1, \ldots, l_n, y_1, \ldots y_p, z_1, \ldots, z_q]$$

– *des paramètres* $\alpha_1, \ldots, \alpha_{m_0+m_1}$ *appartenant à* $\mathrm{par}(D) = \{w_1^{\tau_1}, \ldots, w_{n+p+q}^{\tau_{n+p+q}}\}$,
– *des paramètres* $\alpha'_1, \ldots, \alpha'_{m_0} \in \mathrm{par}(C)$, *et*
– *des contextes publics* $D_0, \ldots, D_{m_1}$

tels que

(i) $D = D_0[\alpha_1, \ldots, \alpha_{m_0+m_1}]$ *avec*

$$\begin{aligned} \{\alpha_1, \ldots, \alpha_{m_0}\} &= \{w_1^{\tau_1}, \ldots, w_n^{\tau_n}\} \\ \{\alpha_{m_0+1}, \ldots, \alpha_{m_0+m_1}\} &= \{w_{n+1}^{\tau_{n+1}}, \ldots, w_{n+p+q}^{\tau_{n+p+q}}\} \end{aligned}$$

(ii) $C = D_0[\alpha'_1, \ldots, \alpha'_{m_0}, D_1, \ldots, D_{m_1}]$.

En particulier,
– *si les* $t_1, \ldots, t_m$ *sont* $\mathcal{R}$*-réduits,* D *est propre ;*
– *pour tout* l_i*, il existe un* j *tel que* $l_i\sigma = t_j$ *; et*
– *pour tout* y_i *(resp.* z_j*), il existe un* k *tel que* $y_i\sigma = D_k[t_1, \ldots, t_n]$ *(resp.* $z_j\sigma = D_k[t_1, \ldots, t_n]$*).*

Démonstration. Soit $x_1, \ldots, x_n$ des variables fraîches telles que $\mathrm{sort}(x_i) = \mathrm{sort}(t_i)$. La substitution $\mu = \sigma\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ étant un unificateur de $C[x_1, \ldots, x_n]$ et de l, il existe un plus grand contexte commun D_0 à ces deux termes de sorte que :

$$\begin{aligned} l &= D_0[l'_1, \ldots, l'_{m_0}, x'_1, \ldots, x'_{m_1}] \\ C[x_1, \ldots, x_n] &= D_0[x''_1, \ldots, x''_{m_0}, D_1[x_1, \ldots, x_n], \ldots, D_{m_1}[x_1, \ldots, x_n]] \end{aligned}$$

où les termes l'_i ne sont pas des variables et D_0 utilise tous ses paramètres (en particulier $x'_i \in \mathrm{var}(l)$ et $x''_j \in \{x_1, \ldots, x_n\}$).

On obtient la décomposition voulue $l = D[l_1, \ldots, l_n, y_1, \ldots y_p, z_1, \ldots, z_q]$ en regroupant dans l'expression $l = D_0[l'_1, \ldots, l'_{m_0}, x'_1, \ldots, x'_{m_1}]$ les termes identiques parmi $l'_1, \ldots, l'_{m_0}$ d'une part, et parmi $x'_1, \ldots, x'_{m_1}$ d'autre part ; puis en réordonnant les paramètres de D de manière à faire apparaître en premier les variables $y_i \in \mathrm{var}(l'_1, \ldots, l'_{m_0}) = \mathrm{var}(l_1, \ldots, l_n)$. □

Autres notations

Rappelons que $k_0 \triangleq \max\{\mathrm{ar}(Y) \mid Y \in \mathcal{Y}\}$ désigne le rang maximal des paramètres et de l'arité des variables du second ordre du problème.

Dans la suite, nous fixons un symbole spécial $\perp$ distinct des autres valeurs du calcul. Nous fixons également une constante publique c_τ de sorte τ pour chaque τ dans la signature.

De plus, nous notons $\mathrm{st}_{\neq}(t) \triangleq \mathrm{st}(t) - \{t\}$ l'ensemble des sous-termes stricts d'un terme t. Dans le cas d'un ensemble ou d'un n-uplet de termes, nous posons de même $\mathrm{st}_{\neq}(t_1, \ldots, t_n) \triangleq \mathrm{st}(t_1, \ldots, t_n) - \{t_1, \ldots, t_n\}$.

En accord avec le chapitre 2, la notation $\mathrm{mgu}(t_1, t_2)$ désigne un unificateur principal de t_1 et t_2 lorsqu'il existe ; dans le cas contraire, on pose $\mathrm{mgu}(t_1, t_2) = \perp$. Dans la suite, nous supposons que l'algorithme d'unification est choisi de manière à ne jamais créer de variables[2] : $\mathrm{var}(\mathrm{mgu}(t_1, t_2)) \subseteq \mathrm{var}(t_1, t_2)$. Par commodité,

[2]Bien que tous les algorithmes d'unification usuels vérifient cette condition, la notion d'unificateur principal n'exclut pas cette possibilité en général.

nous supposons également que cet algorithme substitue en priorité les variables de son second argument. Plus précisément, pour tous termes t et l tels que $\mathrm{var}(t) \cap \mathrm{var}(l) = \emptyset$ et $t = l\sigma$, on demande $\mathrm{mgu}(t, l) = \sigma$. En outre, la notation mgu est généralisée à l'unification de n-uplets de termes de la manière attendue : $\mu = \mathrm{mgu}((t_1, \ldots, t_n), (t'_1, \ldots, t'_n))$ unifie simultanément chaque t_i avec t'_i.

Étant donnés un ensemble de règles Φ et deux termes du premier ordre t_0 et r, on note

$$\mathrm{st}^1(\Phi) \triangleq \{t \mid M \rhd t \in \Phi\} \qquad \Phi\{t_0 \mapsto r\} \triangleq \{M \rhd t\{t_0 \mapsto r\} \mid (M \rhd t) \in \Phi\}$$

et l'on définit de manière similaire $\mathrm{st}^1(\mathcal{C})$ et $\mathcal{C}\{t_0 \mapsto r\}$ pour tout ensemble de contraintes $\mathcal{C}$.

Nous introduisons enfin une relation entre termes du second ordre, notée $\preceq$, que nous appelons *relation (ou pré-ordre) de stratification*, pour des raisons qui apparaîtront dans les sections suivantes.

Définition 4.6. La relation de stratification $\preceq$ est définie par $M_1 \preceq M_2$ ssi
- $\mathrm{maxpar}(M_1) \leq \mathrm{maxpar}(M_2)$, et
- pour tout $Y \in \mathrm{var}(M_1)$, ou bien $Y \in \mathrm{var}(M_2)$ ou bien $\mathrm{ar}(Y) < \mathrm{maxpar}(M_2)$.

Notez que dans le cas présent, on ne demande pas que M_1 et M_2 soient de même sorte. Nous détaillons plus loin (section 4.3) plusieurs propriétés utiles de la relation de stratification $\preceq$, notamment le fait d'être un pré-ordre (partiel), stable par application de contextes et de substitutions bien formées. Intuitivement, dans le cas où M_1 et M_2 sont deux recettes possibles pour un même terme t, la relation $M_1 \preceq M_2$ signifie que M_1 est peut-être utilisé « partout » à la place de M_2.

Composition des substitutions idempotentes

Dans la suite nous utilisons le lemme technique suivant pour justifier la bonne formation syntaxique des systèmes de contraintes lors de la procédure.

Lemme 4.7. *Soit μ et μ' des substitutions idempotentes telles que* $\mathrm{supp}(\mu')\mu' \cap \mathrm{supp}(\mu) = \emptyset$. *Alors $\mu\mu'$ est idempotente et* $\mathrm{supp}(\mu\mu') = \mathrm{supp}(\mu) \cup \mathrm{supp}(\mu')$.

Démonstration. Montrons que $V\mu'\mu\mu' = V\mu'$ pour tout V dans $\mathrm{var}(\mathrm{im}(\mu))$ (de sorte que $\mu\mu'\mu\mu' = \mu\mu'$). Si $V \notin \mathrm{supp}(\mu')$, alors $V\mu'\mu\mu' = V\mu\mu' = V\mu'$ car μ est idempotente.

Sinon $V \in \mathrm{supp}(\mu')$ implique $V\mu' \in \mathrm{supp}(\mu')\mu' \subseteq \mathcal{X} - \mathrm{supp}(\mu)$, et par conséquent $V\mu'\mu\mu' = V\mu'^2 = V\mu'$.

Par ailleurs, $\mathrm{supp}(\mu\mu') \subseteq \mathrm{supp}(\mu) \cup \mathrm{supp}(\mu')$ est clair.

Réciproquement, on a d'une part $\mathrm{supp}(\mu') - \mathrm{supp}(\mu) \subseteq \mathrm{supp}(\mu\mu')$. D'autre part, supposons qu'il existe $V \in \mathrm{supp}(\mu)$ tel que $V\mu\mu' = V$. Comme $V\mu \neq V$, on a $V\mu\mu' = V \neq V\mu$, soit $V\mu \in \mathrm{supp}(\mu')$. Par conséquent, $V = V\mu\mu' \in \mathrm{supp}(\mu')\mu' \subseteq \mathcal{X} - \mathrm{supp}(\mu)$; contradiction. Ainsi, $\mathrm{supp}(\mu) \subseteq \mathrm{supp}(\mu\mu')$, et finalement $\mathrm{supp}(\mu\mu') = \mathrm{supp}(\mu) \cup \mathrm{supp}(\mu')$. □

Règles de transformation

Nous considérons maintenant la relation de transformation $\Longrightarrow$ engendrée par les règles des tables 4.1 et 4.2.

Project

$$\frac{\begin{array}{ll}(M \rhd t) \in \Phi & \mu = \mathrm{mgu}(t, f(t_1, \ldots, t_n)) \\ X \notin \mathrm{var}(M) & \mathrm{maxpar}(M), \mathrm{maxar}(M) \leq \mathrm{ar}(X)\end{array}}{\Phi;\, \Psi;\, \mathcal{C} \uplus \{X \rhd^? f(t_1, \ldots, t_n)\};\, \mathcal{N};\, \sigma \Longrightarrow (\, \Phi;\, \Psi;\, \mathcal{C};\, \mathcal{N};\, \sigma\,)\,\{X \mapsto M\}\mu}$$

Imitate

$$\frac{f \in \mathcal{F}_{\mathrm{pub}} \qquad X_1, \ldots, X_n \text{ variables fraîches d'arité } \mathrm{ar}(X_i) = \mathrm{ar}(X)}{\begin{array}{l}\Phi;\, \Psi;\, \mathcal{C} \uplus \{X \rhd^? f(t_1, \ldots, t_n)\};\, \mathcal{N};\, \sigma \\ \Longrightarrow (\, \Phi;\, \Psi;\, \mathcal{C} \cup \{X_1 \rhd^? t_1, \ldots, X_n \rhd^? t_n\};\, \mathcal{N};\, \sigma\,)\,\{X \mapsto f(X_1, \ldots, X_n)\}\end{array}}$$

Coalesce

$$\frac{\mathrm{ar}(X_1) \leq \mathrm{ar}(X_2)}{\Phi;\, \Psi;\, \mathcal{C} \uplus \{X_1 \rhd^? t,\, X_2 \rhd^? t\};\, \mathcal{N};\, \sigma \Longrightarrow (\, \Phi;\, \Psi;\, \mathcal{C} \cup \{X_1 \rhd^? t\};\, \mathcal{N};\, \sigma\,)\,\{X_2 \mapsto X_1\}}$$

Fail

$$\frac{\begin{array}{l}(X \rhd^? t_0) \in \mathcal{C} \qquad t_0 = f(t_1, \ldots, t_n) \qquad f \notin \mathcal{F}_{\mathrm{pub}} \\ \text{pour tout } (M \rhd t) \in \Phi, \text{ si } \left\{\begin{array}{l} X \notin \mathrm{var}(M) \\ \mathrm{maxpar}(M) \leq \mathrm{ar}(X) \text{ alors } \mathrm{mgu}(t, t_0) = \bot \\ \mathrm{maxar}(M) \leq \mathrm{ar}(X)\end{array}\right.\end{array}}{\Phi;\, \Psi;\, \mathcal{C};\, \mathcal{N};\, \sigma \Longrightarrow \bot}$$

Tab. 4.1 – Règles de pré-résolution

Narrowing

$$\frac{\begin{array}{l} l \to r \text{ règle fraîche issue de } \mathcal{R} \\ t_0 = f(t_1, \ldots, t_n) \in \mathrm{st}^1(\Phi, \mathcal{C}) - \mathrm{st}(\mathcal{N}) - \mathcal{X} \qquad \mu = \mathrm{mgu}(t_0, l) \end{array}}{\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \Longrightarrow (\ \Phi\{t_0 \mapsto r\};\ \Psi;\ \mathcal{C}\{t_0 \mapsto r\};\ \mathcal{N} \cup \{t_1, \ldots, t_n\};\ \sigma\ )\,\mu}$$

Constrain

$$\frac{\mu = \mathrm{mgu}(t, t')}{\Phi;\ \Psi;\ \mathcal{C} \uplus \{t =^?_E t'\};\ \mathcal{N};\ \sigma \Longrightarrow (\ \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N} \cup \{t\};\ \sigma\ )\,\mu}$$

Context-1

$$\frac{\begin{array}{l} l \to r \text{ règle fraîche issue de } \mathcal{R} \\ l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q] \text{ décomposition propre} \\ \qquad \text{telle que } r \in \mathrm{st}_{\neq}(l_1, \ldots, l_n) \cup \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}} \\ (M_1 \rhd t_1), \ldots, (M_n \rhd t_n) \in \Phi \qquad \mu = \mathrm{mgu}(\ (t_1, \ldots, t_n),\ (l_1, \ldots, l_n)\ ) \\ Y_1, \ldots, Y_p,\ Z_1, \ldots, Z_q \text{ variables fraîches d'arité } k \in \{0, 1, \ldots, k_0\} \\ M = D[M_1, \ldots, M_n, Y_1, \ldots, Y_p, Z_1, \ldots, Z_q] \\ M' = M\{Z_j \mapsto \mathsf{c}_{\mathrm{sort}(Z_j)}\}_{1 \le j \le q} \end{array}}{\begin{array}{l} \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \\ \Longrightarrow (\Phi \cup \{M' \rhd r\};\ \Psi \cup \{\forall Z_1, \ldots, Z_q.M \bowtie M'\}; \\ \qquad \mathcal{C} \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\};\ \mathcal{N} \cup \{l_1, \ldots, l_n\};\ \sigma)\mu \end{array}}$$

Context-2

$$\frac{\begin{array}{l} l \to r \text{ règle fraîche issue de } \mathcal{R} \\ l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q] \text{ décomposition propre} \\ \qquad \text{telle que } r = D'[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q] \\ (M_1 \rhd t_1), \ldots, (M_n \rhd t_n) \in \Phi \qquad \mu = \mathrm{mgu}(\ (t_1, \ldots, t_n),\ (l_1, \ldots, l_n)\ ) \\ Y_1, \ldots, Y_p,\ Z_1, \ldots, Z_q \text{ variables fraîches d'arité } k \in \{0, 1, \ldots, k_0\} \\ M = D[M_1, \ldots, M_n, Y_1, \ldots, Y_p, Z_1, \ldots, Z_q] \\ M' = D'[M_1, \ldots, M_n, Y_1, \ldots, Y_p, Z_1, \ldots, Z_q] \end{array}}{\begin{array}{l} \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \\ \Longrightarrow (\Phi;\ \Psi \cup \{\forall Z_1, \ldots, Z_q.M \bowtie M'\}; \\ \qquad \mathcal{C} \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\};\ \mathcal{N} \cup \{l_1, \ldots, l_n\};\ \sigma)\mu \end{array}}$$

Relate

$$\frac{(M \rhd t) \in \Phi \qquad X \text{ variable fraîche d'arité } k_0}{\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \Longrightarrow \Phi;\ \Psi \cup \{X \bowtie M\};\ \mathcal{C} \cup \{X \rhd^? t\};\ \mathcal{N} \cup \{t\};\ \sigma}$$

Discard

$$\frac{N \rhd_{\Phi,\mathcal{C}} t \qquad N \preceq M}{\Phi \uplus \{M \rhd t\};\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \Longrightarrow \Phi;\ \Psi \cup \{N \bowtie M\};\ \mathcal{C};\ \mathcal{N};\ \sigma}$$

TAB. 4.2 – Règles principales

Formellement, par *variable fraîche*, nous entendons une variable de sorte appropriée (ici toujours fixée par le contexte), n'apparaissant pas dans les variables $\mathrm{var}(\Sigma)$ du système de départ de la règle considérée $\Sigma \Longrightarrow \Sigma'$. Un *renommage frais* ρ est un renommage bien formé dont l'image est constituée de variables fraîches deux à deux distinctes. Une *règle fraîche issue de* $\mathcal{R}$ est une règle de la forme $l\rho \to r\rho$ où $(l \to r) \in \mathcal{R}$ où ρ est un renommage frais de domaine $\mathrm{var}(l, r) = \mathrm{var}(r)$ ($\mathcal{R}$ étant convergent).

Les règles de transformations (excepté **Fail**) préservent bien la syntaxe des systèmes de contraintes $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$:

- d'une part, les conditions de fraîcheur dans les règles **Imitate**, **Context-{1,2}** et **Relate** garantissent la condition : pour tout $X \rhd^? t \in \mathcal{C}$, X apparaît une seule fois dans $\mathcal{C}$;
- d'autre part, les substitutions utilisées sont toujours bien formées, idempotentes et appliquées successivement au système entier, de sorte que par le lemme 4.7, la condition $\Sigma\sigma = \sigma$ est bien préservée.

Les règles de transformation se répartissent en deux groupes en fonction de leur rôle dans la procédure :

1. Les *règles de pré-résolution* (table 4.1) ont pour but de simplifier les systèmes de contraintes pour les amener, sauf cas d'échec, dans une forme pré-résolue.
2. Les *règles principales* (table 4.2) constituent le coeur de l'algorithme. Nous discutons plus bas une stratégie d'application de ces règles permettant la terminaison du système.

Nous commentons maintenant le rôle de chaque règle plus précisément.

1. Parmi les règles de pré-résolution (table 4.1), la règle **Project** résout une contrainte de déductibilité de $\mathcal{C}$ à l'aide d'une règle d'environnement dans Φ.

   La règle **Imitate** permet de décomposer une contrainte de déductibilité en contraintes plus petites lorsque le symbole en position de tête f est public.

   La règle **Coalesce** fusionne deux contraintes portant sur le même terme du premier ordre. Tandis que les règles **Project** et **Imitate** s'inspirent des règles classiques du même nom en unification de second ordre (voir par exemple Huet [Hue75]), nous verrons que la complétude de la règle **Coalesce** repose sur plusieurs propriétés spécifiques des systèmes étudiés.

   La règle **Fail** conclut $\bot$ (« faux ») lorsqu'aucune des règles précédentes ne s'applique. Ce cas de figure est possible dans un cadre cryptographique car la signature $\mathcal{F}$ contient typiquement des symboles non publics (noms, constantes privées...).

   Le lemme suivant justifie finalement le nom des règles de pré-résolution.

   Lemme 4.8. *Un système de contraintes* $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ *est pré-résolu ss'il est saturé pour les règles de pré-résolution.*

   Démonstration. Si Σ est pré-résolu, par définition $\mathcal{C}$ est de la forme

   $$\mathcal{C} = \{X_1 \rhd^? x_1, \ldots, X_n \rhd^? x_1\}$$

   où les variables $x_1, \ldots, x_n$ (tout comme les $X_1, \ldots, X_n$) sont deux à deux disjointes. Clairement les règles de pré-résolution ne s'appliquent pas à Σ.

Réciproquement, si Σ n'est pas pré-résolu, alors : ou bien il existe $(X \rhd^? f(t_1, \ldots, t_n)) \in \mathcal{C}$ et l'une des trois règles **Project**, **Imitate** ou **Fail** s'applique ; ou bien il existe deux contraintes $X_1 \rhd^? x$ et $X_2 \rhd^? x$ $(X_1 \neq X_2)$ dans $\mathcal{C}$, si bien que la règle **Coalesce** s'applique. □

2. En ce qui concerne les règles principales (table 4.2), les règles **Narrowing** et **Constrain** s'inspirent de la notion classique de surréduction ou *narrowing* (voir par exemple [JK91, Hul80, DJ04a]). La première règle **Narrowing** a pour but de deviner les réductions possibles dans les termes connus (Φ) et dans les contraintes ($\mathcal{C}$) d'un système. Inversement, la seconde règle **Constrain** résout une contrainte d'égalité par unification syntaxique.

   Afin de rendre la terminaison possible, nous nous inspirons d'une restriction de la surréduction appelée *surréduction basique* (*basic narrowing*) connue pour rester complète en présence de systèmes de réécriture convergent [Hul80, MH94, DJ04a]).

   Plus précisément, nous utilisons la composante $\mathcal{N}$ des systèmes de contraintes pour enregistrer les termes pour lesquels la surréduction est terminée. À chaque étape de surréduction sur un terme $t_0 = f(t_1, \ldots, t_n)$ par la règle **Narrowing**, les sous-termes $t_1, \ldots, t_n$ sont indiqués comme étant surréduits. Ceci correspond au fait que dans un système convergent il est possible choisir une stratégie de réduction en profondeur d'abord (*innermost*).

   Notons que dans le cas d'une règle sous-terme $l \to r$, si $\mu = \mathrm{mgu}(t_0, l)$ alors $r\mu$ est ou bien un sous-terme de $l_1\mu, \ldots, l_n\mu$ ou bien un terme clos $\mathcal{R}$-réduit. Dans les deux cas, on observe que la règle de surréduction ne pourra plus s'appliquer en dessous (au sens large) des positions de t_0 dans Φ et $\mathcal{C}$.

   Les deux règles **Context-{1,2}**, plus complexes, rendent compte des réductions apparaissant dans des contextes de termes connus. La différence entre les deux règles réside dans la génération ou non d'une nouvelle règle d'environnement selon que

   (a) r est un terme clos R-réduit ou un sous-terme strict de $l_1, \ldots, l_n$, ou que

   (b) il existe un contexte public D' tel que

   $$r = D'[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q].$$

   On remarque que toute décomposition D d'une règle sous-terme $l \to r$ de $\mathcal{R}$ est de type (a) ou de type (b).

   L'ajout de $l_1, \ldots, l_n$ dans $\mathcal{N}$ à l'issue des règles **Context-{1,2}** correspond, comme **Narrowing**, à une stratégie de réduction en profondeur d'abord justifiée par la preuve de complétude.

   Notons que l'arité des variables fraîches $Y_1, \ldots, Y_p$, $Z_1, \ldots, Z_q$ est une valeur $k \in \{0, \ldots, k_0\}$, laissée libre bien que la valeur $k = k_0$ subsume intuitivement toutes les autres. Ce choix technique vise à faciliter la preuve de complétude.

   Nous détaillons plus bas différents exemples d'application des règles **Context-{1,2}**.

   La règle **Relate** a pour but de collecter de nouvelles équations visibles entre termes connus. Nous verrons que cette opération est cruciale pour assurer la complétude de la règle **Coalesce**.

   Enfin, la règle **Discard** est une variante de **Relate** utilisée pour supprimer les règles $M \rhd t$ intuitivement subsumées par des règles d'environnement et/ou

des contraintes de déductibilité existantes. La suppression d'une règle se fait au prix de l'ajout d'une nouvelle équation dans Ψ. La condition technique $N \preceq M$ vise à garantir la complétude lorsque cette règle est appliquée de manière prioritaire.

Exemple 4.3. Considérons à nouveau la règle du chiffrement symétrique déterministe $(l \to r) = (\mathsf{dec}(\mathsf{enc}(x, y), y) \to x)$ (exemple 4.1). La seconde décomposition propre de cette règle :

$$l = D_2[\mathsf{enc}(x, y), y] \text{ où } D_2 = \mathsf{dec}(w_1, w_2)$$

satisfait $r = x \in \mathrm{st}_{\neq}(\mathsf{enc}(x, y))$. Elle conduit donc à instancier la règle **Context-1** de la manière suivante :

$$\frac{(M_1 \rhd t_1) \in \Phi \qquad \mu = \mathrm{mgu}(t_1, \mathsf{enc}(x_1, y_1)) \qquad Y_1, x_1, y_1 \text{ variables fraîches}}{\begin{array}{l}\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \\ \Longrightarrow (\Phi \cup \{\mathsf{dec}(M_1, Y_1) \rhd x_1\};\ \Psi';\ \mathcal{C} \cup \{Y_1 \rhd^? y_1\};\ \mathcal{N} \cup \{\mathsf{enc}(x_1, y_1)\};\ \sigma)\mu\end{array}}$$

où $\Psi' = \Psi \cup \{\mathsf{dec}(M_1, Y_1) \bowtie \mathsf{dec}(M_1, Y_1)\}$ est ici équivalent à Ψ. Si l'on résout l'unification dans la règle ci-dessus, on obtient encore :

$$\frac{(M_1 \rhd \mathsf{enc}(s_1, s_2)) \in \Phi \qquad Y_1 \text{ variable fraîche}}{\begin{array}{l}\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \\ \Longrightarrow \Phi \cup \{\mathsf{dec}(M_1, Y_1) \rhd s_1\};\ \Psi';\ \mathcal{C} \cup \{Y_1 \rhd^? s_2\};\ \mathcal{N} \cup \{\mathsf{enc}(s_1, s_2)\};\ \sigma\end{array}}$$

Autrement dit, si l'on connaît un terme $\mathsf{enc}(s_1, s_2)$, alors on peut déduire s_1 pourvu que la clé s_2 soit connue.

Exemple 4.4. Considérons encore la règle du chiffrement asymétrique probabiliste $(l \to r) = (\mathsf{pdec}(\mathsf{penc}(x, \mathsf{pub}(y), z), y) \to x)$ (exemple 4.2), et la deuxième décomposition obtenue :

$$l = D_2[\mathsf{pub}(y), y, x, z] \text{ où } D_2 = \mathsf{pdec}(\mathsf{penc}(w_3, w_1, w_4), w_2)$$

La décomposition D_2 satisfait $r = x = w_3[\mathsf{pub}(y), y, x, z]$. Ceci conduit donc à l'instanciation suivante de la règle **Context-2** :

$$\frac{\begin{array}{l}(M_1 \rhd t_1) \in \Phi \qquad \mu = \mathrm{mgu}(t_1, \mathsf{pub}(y_1)) \\ Y_1, Z_1, Z_2, y_1 \text{ variables fraîches} \qquad (Y_1, Z_1, Z_2 \text{ d'arité } k)\end{array}}{\begin{array}{l}\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \\ \Longrightarrow (\Phi;\ \Psi \cup \{\forall Z_1, Z_2.\mathsf{pdec}(\mathsf{penc}(Z_1, M_1, Z_2), Y_1) \bowtie Z_1\}; \\ \quad \mathcal{C} \cup \{Y_1 \rhd^? y_1\};\ \mathcal{N} \cup \{\mathsf{pub}(y)\};\ \sigma)\mu\end{array}}$$

ou encore, en résolvant l'unification :

$$\frac{(M_1 \rhd \mathsf{pub}(s_1) \in \Phi \qquad Y_1, Z_1, Z_2 \text{ variables fraîches d'arité } k}{\begin{array}{l}\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \\ \Longrightarrow \Phi;\ \Psi \cup \{\forall Z_1, Z_2.\mathsf{pdec}(\mathsf{penc}(Z_1, M_1, Z_2), Y_1) \bowtie Z_1\}; \\ \quad \mathcal{C} \cup \{Y_1 \rhd^? s_1\};\ \mathcal{N} \cup \{\mathsf{pub}(s_1)\};\ \sigma\end{array}}$$

Autrement dit, si l'on connaît une clé privée s_1, on peut reconnaître la clé publique correspondante $\mathsf{pub}(s_1)$ en testant en chiffrant un nombre arbitraire Z_1 et testant le déchiffrement.

Si, comme on peut s'y attendre, le symbole pub est public, une autre manière de reconnaître $\mathsf{pub}(s_1)$ sachant s_1 est simplement d'appliquer le symbole pub. Ceci est visible par la séquence suivante, utilisant la règle **Relate**. Si $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ et $(M_1 \rhd \mathsf{pub}(s_1)) \in \Phi$, on a :

$$\begin{aligned}\Sigma \quad &\Longrightarrow_{\mathbf{Relate}} \quad \Phi;\ \Psi \cup \{X \bowtie M_1\};\ \mathcal{C} \cup \{X \rhd^? \mathsf{pub}(s_1)\};\\ &\qquad\qquad \mathcal{N} \cup \{\mathsf{pub}(s_1)\};\ \sigma\\ &\Longrightarrow_{\mathbf{Imitate}} \quad \Phi;\ \Psi \cup \{\mathsf{pub}(Y) \bowtie M_1\};\ \mathcal{C} \cup \{Y \rhd^? s_1)\};\\ &\qquad\qquad \mathcal{N} \cup \{\mathsf{pub}(s_1)\};\ \sigma\{X \mapsto \mathsf{pub}(Y)\}\end{aligned}$$

Dérivations standard

Pour différentes raisons — notamment les règles **Context-{1,2}** et **Relate** — la relation $\Longrightarrow$ définie plus haut ne termine pas au sens usuel (*i.e.* l'absence de chaînes infinies).

Nous présentons ici différentes restrictions sur les conditions d'application des règles permettant d'assurer la terminaison de la procédure générale décrite dans la sous-section 4.1.2, dans le cas d'un système de réécriture $\mathcal{R}$ sous-terme-convergent et fini. Naturellement il conviendra de vérifier la complétude des dérivations en tenant compte de ces restrictions.

Afin de formaliser la notion de terminaison et de branchement fini des règles de transformation, nous introduisons tout d'abord les notions suivantes d'équivalence et de subsomption entre systèmes.

Définition 4.7. L'ensemble des *variables non résolues* de $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ est défini par $\mathrm{var}^*(\Sigma) \triangleq \mathrm{var}(\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \mathcal{Y}\sigma)$.

Deux systèmes de contraintes $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ et $\Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma'$ sont *équivalents modulo renommage*, ce que l'on note $\Sigma \equiv \Sigma'$, ss'il existe une substitution ρ telle que $\rho|_{\mathrm{var}^*(\Sigma)}$ est un renommage bien formé inversible, telle que

$$(\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma|_{\mathcal{Y}})\rho \ = \ \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma'|_{\mathcal{Y}}$$

et telle que si Σ ou Σ' est standard, on a $\{X^*, Y^*\} \cap \mathrm{supp}(\rho) = \emptyset$.

Enfin, Σ *subsume* Σ', ce que l'on note $\Sigma \geq \Sigma'$, ss'il existe une substitution ρ tel que $\rho|_{\mathrm{var}^*(\Sigma)}$ est un renommage bien formé inversible et

$$\Phi\rho = \Phi',\ \Psi\rho = \Psi',\ \mathcal{C}\rho = \mathcal{C}',\ \mathcal{N}\rho \subseteq \mathcal{N}',\ \sigma\rho = \sigma'$$

Intuivement, un système Σ' est donc subsumé par Σ lorsque, modulo renommage, $\mathcal{N}'$ est plus grand que $\mathcal{N}$, et σ' est égale à σ augmentée d'une substitution portant sur des variables n'apparaissant pas dans $\mathrm{var}^*(\Sigma)$. Les propriétés suivantes découlent facilement des définitions des systèmes de contraintes et des règles :

Proposition 4.9. *Supposons $\Sigma \equiv \Sigma'$. Alors*

(1) Σ est résolu (respectivement standard, pré-résolu) ssi Σ' l'est ;

(2) Σ et Σ' ont les mêmes solutions ;

(3) si $\Sigma \Longrightarrow \Sigma_1$ par une règle de transformation $\mathbf{R}$, alors il existe Σ'_1 tel que $\Sigma_1 \equiv \Sigma'_1$ et $\Sigma' \Longrightarrow \Sigma'_1$ par la même règle.

Supposons $\Sigma \geq \Sigma'$. Alors

(4) si Σ' est résolu, alors Σ est résolu ;

(5) toute solution de Σ' est solution de Σ ;

(6) si $\Sigma' \Longrightarrow \Sigma'_1$ par une règle de transformation $\mathbf{R}$, alors il existe Σ_1 tel que $\Sigma'_1 \geq \Sigma_1$ et $\Sigma \Longrightarrow \Sigma_1$ par la même règle.

Lors de la procédure, nous pouvons donc considérer modulo $\equiv$ les successeurs d'un système de contraintes. Dans le cas où $\mathcal{R}$ — et par conséquent le nombre de décompositions de règles dans $\mathcal{R}$ — est fini, ceci garantit la propriété de branchement fini des règles de transformation.

De plus, nous utilisons la relation $\geq$ comme condition d'arrêt des dérivations : en utilisant la proposition 4.9 nous vérifions en effet plus loin que toute dérivation $\Sigma_0 \Longrightarrow^* \Sigma_1$ vérifiant $\theta \models \Sigma_0$ et $\theta \models \Sigma_1$ peut être transformée en une dérivation vérifiant ces mêmes hypothèses et dans laquelle, si Σ apparaît avant Σ', alors $\Sigma \not\geq \Sigma'$.

Notons que $\geq$ est décidable en temps polynomial dans la taille des systèmes considérés : en effet, les valeurs de ρ sont déterminées par σ et σ' ; il suffit ensuite de vérifier les autres conditions.

Considérons l'ordre (partiel) strict entre les règles $>$, appelé *ordre de priorité* et engendré par les relations suivantes :

$$\textbf{Coalesce} > \textbf{Project} > \textbf{Fail}$$
$$\textbf{Coalesce} > \textbf{Imitate} > \textbf{Fail}$$
$$\textbf{Fail} > \textbf{Discard}$$
$$\textbf{Fail} > \textbf{Discard}$$
$$\textbf{Discard} > \textbf{Narrowing}$$
$$\textbf{Discard} > \textbf{Constrain}$$
$$\textbf{Discard} > \textbf{Context-1}$$
$$\textbf{Discard} > \textbf{Context-2}$$
$$\textbf{Discard} > \textbf{Relate}$$

Définition 4.8. Soit Σ_0 un système initial et

$$\Sigma_0 \Longrightarrow_{\mathbf{R}_1} \Sigma_1 \Longrightarrow_{\mathbf{R}_2} \Sigma_2 \ldots \Longrightarrow_{\mathbf{R}_n} \Sigma_n$$

une séquence de transformations (aussi appelée *dérivation*) à partir de Σ_0. On dit qu'une telle séquence est *standard* lorsque les points suivants sont respectés :

(1) pour tout $1 \leq i \leq n$, s'il existe un système Σ'_i et une règle $\mathbf{R}'_i$ telle que $\Sigma_{i-1} \Longrightarrow_{\mathbf{R}'_i} \Sigma'_i$ alors $\mathbf{R}'_i \not> \mathbf{R}_i$; autrement dit, une règle est applicable uniquement sur les systèmes saturés par les règles de priorité supérieure.

(2) pour tout $1 \leq i < j \leq n$, on a $\Sigma_i \not\geq \Sigma_j$ (*condition d'arrêt*) ;

(3) lors de la dérivation, la règle **Discard** est appliquée en priorité aux règles d'environnement $M \rhd t$ les plus récemment ajoutées au système.

Plus généralement, une séquence de transformations *respecte les priorités entre règles* ssi elle vérifie les points (1) et (3) ci-dessus. On remarque que dans une telle dérivation, les règles principales (table 4.2) s'appliquent uniquement à des systèmes de contraintes pré-résolus, du fait du lemme 4.8.

Le dernier point (3) s'avère techniquement nécessaire à la terminaison des règles en général. Formellement, cette condition s'énonce par exemple en considérant Φ comme un ensemble muni d'un ordre total strict $<_\Phi$ représentant l'ordre de création des règles dans Φ. Par souci de simplicité, nous traitons cet ordre de manière implicite dans la suite de la preuve.

Terminaison des règles prioritaires

Nous terminons la description des règles de transformation en vérifiant que les *règles prioritaires* des dérivations standard, à savoir les règles de pré-résolution (table 4.1) et la règle **Discard** (table 4.2), terminent bel et bien.

En effet, concernant les règles de pré-résolution, **Coalesce** et **Imitate** diminuent la taille totale des membres droits de contraintes $X \rhd^? t$ du système, sans créer de variables du premier ordre. **Project** fait décroître le nombre de variables du premier ordre non résolues si $\mathrm{supp}(\mu) \neq \emptyset$, et diminuent la taille des contraintes (dans le sens ci-dessus) dans le cas contraire. Par ailleurs, la règle **Fail** interrompt le calcul par construction.

Quant à la règle **Discard**, appliquée à un système $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$, cette règles fait décroître strictement le cardinal de Φ, tout en laissant le système pré-résolu.

4.1.4 Plan de la preuve

Les sections suivantes constituent la preuve détaillée des propriétés attendues des règles de transformation.

Pour commencer (section 4.2) nous vérifions la correction des règles et la préservation des systèmes standard pour tout système $\mathcal{R}$.

Nous établissons ensuite dans la section 4.3 un certain nombre de résultats préliminaires, portant sur les règles de pré-résolution, le pré-ordre de stratification et différents invariants des systèmes accessibles.

À l'aide de ces résultats, nous montrons la complétude des dérivations standard pour tout système $\mathcal{R}$ sous-terme-convergent dans la section 4.4. Nous montrons enfin la terminaison des règles lorsque $\mathcal{R}$ est de plus fini (section 4.5), et concluons la preuve du théorème 4.1 dans la section 4.6.

4.2 Correction des règles de transformation

Dans cette section, nous montrons la correction des règles de transformation pour une séquence arbitraire (non nécessairement standard) à partir d'un système initial, pour une théorie équationnelle E arbitraire.

Nous énonçons tout d'abord quelques propriétés simples des règles de transformation.

Lemme 4.10. *Soit $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ et $\Sigma' = \Phi'; \Psi'; \mathcal{C}'; \mathcal{N}'; \sigma'$ deux systèmes de contraintes tels que $\Sigma \Longrightarrow^* \Sigma'$. Alors on a*

(1) $\mathrm{var}(\Sigma) \subseteq \mathrm{var}(\Sigma')$,

(2) $\sigma\sigma' = \sigma'$,

(3) $\Psi\sigma' \subseteq \Psi'$ *(en particulier la relation $\bowtie_{\Psi'}$ contient $\bowtie_{\Psi\sigma'}$), et*

(4) $\mathcal{N}\sigma' \subseteq \mathcal{N}'$.

Démonstration. Évident par récurrence sur la longueur de la dérivation et par définition des règles de transformation. □

Nous introduisons ensuite les notions de *solution faible* et de *correction des règles et des équations* d'un système (par rapport à un système initial). Informellement, la notion de solution faible s'obtient à partir de la définition précédente des solutions en considérant tous les termes du premier ordre modulo E — les relations concernant les termes du second ordre restant inchangées.

Définition 4.9. Une substitution partielle close (bien formée) θ de domaine $\mathcal{Y}$ est une *solution faible* de $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ ss'il existe une substitution partielle close (bien formée) λ de domaine $\mathrm{var}(\Sigma)$ telle que

(i) pour chaque contrainte $X \rhd^? t$ dans $\mathcal{C}$, on a $X\lambda \rhd^E_{\Phi\lambda} t\lambda$;

(ii) pour chaque $t_1 =^?_E t_2$ dans $\mathcal{C}$, $t_1\lambda =_E t_2\lambda$;

(iii) pour chaque $t_1 \neq^?_E t_2$ dans $\mathcal{C}$, $t_1\lambda \neq_E t_2\lambda$;

(iv) λ *étend faiblement* σ au sens où pour tout $X \in \mathrm{dom}(\lambda)$, $X\sigma\lambda = \lambda$, et pour tout $x \in \mathrm{dom}(\lambda)$, $x\sigma\lambda =_E \lambda$;

(v) λ est liée à θ sur $\mathcal{Y}$: $\theta \bowtie_{\Psi\lambda} \lambda|_{\mathcal{Y}}$.

On écrit alors $\theta \models^\sharp \Sigma$, ou encore $\theta, \lambda \models^\sharp \Sigma$ pour préciser le λ utilisé.

Définition 4.10. Soit $\Phi_0 = \{w_1 \rhd t_1, \ldots, w_{k_0} \rhd t_{k_0}\}$ un ensemble de règles d'environnement (initiales). Rappelons que pour tout M, $M[\Phi_0]$ désigne le terme

$$M[t_1, \ldots, t_{k_0}] = M\{w_1 \mapsto t_1, \ldots, w_{k_0} \mapsto t_{k_0}\}.$$

Soit $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ un système tel que $\mathrm{var}(\Phi_0) \subseteq \mathrm{var}(\Sigma)$. Les règles et les équations de Σ sont *correctes par rapport à* Φ_0 (ou Φ_0-*correctes*) ssi pour tout $\theta, \lambda \models^\sharp \Sigma$, les conditions respectives suivantes sont vérifiées :

(a) pour tout $M \rhd t$ dans $\Phi\lambda$, $M[\Phi_0\lambda] =_E t$;

(b) pour tout $\forall\beta.M \bowtie N$ dans $\Psi\lambda$, $M[\Phi_0\lambda] =_E N[\Phi_0\lambda]$.

Le lemme suivant établit plusieurs propriétés des solutions faibles utiles pour la preuve de correction.

Lemme 4.11. *Soit* Σ *un système de contraintes.*

(1) $\theta, \lambda \models \Sigma$ *implique* $\theta, \lambda \models^\sharp \Sigma$.

(2) Si Σ *est initial et si* $\theta, \lambda \models^\sharp \Sigma$ *alors il existe* $\lambda' =_E \lambda$ *tel que* $\theta, \lambda' \models \Sigma$.

(3) Supposons Σ *initial et soit* Φ_0 *l'ensemble des règles d'environnement de* Σ. *Soit* Ψ *un ensemble d'équations closes.*
Supposons que $\theta, \lambda \models^\sharp \Sigma$, *que* $\theta' \bowtie_\Psi \theta$, *et que pour tout* $\forall\beta.M \bowtie N$ *dans* Ψ, $M[\Phi_0\lambda] =_E N[\Phi_0\lambda]$. *Alors on a*

$$\theta', \lambda' \models^\sharp \Sigma$$

où $\lambda' = \theta' \uplus \lambda|_{\mathrm{var}^1(\Sigma)}$ *coïncide avec* λ *au premier ordre et avec* θ' *au second ordre.*

Démonstration. (1) Évident d'après les définitions.

(2) Supposons $\theta, \lambda \models^\sharp \Sigma$. Soit $\Sigma = \Phi; \emptyset; \mathcal{C}; \emptyset; \{\}$ où

$$\begin{aligned}\Phi &= \{w_1 \rhd t_1, \ldots, w_{k_0} \rhd t_{k_0}\}\\ \mathcal{C} &= \{X_1 \rhd^? x_1, \ldots, X_m \rhd^? x_m,\ s_1 \sim_1 s'_1, \ldots, s_n \sim_n s'_n\}\\ \mathcal{Y} &= \{X_1, \ldots, X_m\}\end{aligned}$$

et pour tout $1 \leq i \leq n$, $\sim_i \in \{=^?_E, \neq^?_E\}$. Supposons de plus $0 \leq \mathrm{ar}(X_1) \leq \ldots \leq \mathrm{ar}(X_m) \leq k_0$ et les conditions de régularité des systèmes initiaux vérifiées :

(1) les variables $X_1, \ldots, X_m$ et $x_1 \ldots x_m$ sont deux à deux disjointes, et $\mathrm{var}(\Sigma) = \{X_1, \ldots, X_m, x_1, \ldots, x_m\}$;

(2) pour tout $1 \leq i \leq m$, $\mathrm{var}(t_1, \ldots, t_{\mathrm{ar}(X_i)}) \subseteq \{x_1, \ldots, x_{i-1}\}$.

Par hypothèse, on a $x_i\lambda =_E (X_i\lambda)[t_1\lambda, \ldots, t_{\mathrm{ar}(X_i)}\lambda]$.

Étant données les conditions de régularité, soit λ' défini par induction sur i tel que $\lambda'|_{\mathcal{Y}} = \lambda$ et

$$x_i\lambda' = (X_i\lambda')[t_1\lambda', \ldots, t_{\mathrm{ar}(X_i)}\lambda'].$$

Par récurrence sur i, pour tout i, on a $x_i\lambda' =_E x_i\lambda$, d'où $\theta, \lambda' \models \Sigma_0$ et $\lambda' =_E \lambda$.

(3) Reprenons les notations ci-dessus pour Σ, et supposons $\theta, \lambda \models^\sharp \Sigma$: pour tout i, on a $x_i\lambda =_E (X_i\lambda)[t_1\lambda, \ldots, t_{\mathrm{ar}(X_i)}\lambda]$ et $\theta =_E \lambda|_{\mathcal{Y}}$. Sachant que $\theta' \bowtie_\Psi \theta$, et que les équations de Ψ sont valides dans $\Phi_0\lambda$, le lemme 4.2 et l'équation précédente nous donne :

$$x_i\lambda =_E (X_i\theta')[t_1\lambda, \ldots, t_{\mathrm{ar}(X_i)}\lambda];$$

c'est-à-dire $\theta', \lambda' \models^\sharp \Sigma$ si $\lambda' = \theta' \uplus \lambda|_{\mathrm{var}^1(\Sigma)}$. □

Nous pouvons maintenant établir la correction des règles de transformation pour une séquence arbitraire à partir d'un système initial.

Proposition 4.12 (Correction)**.** *Soit $\Sigma_0 = \Phi_0; \emptyset; \mathcal{C}_0; \emptyset; \{\}$ un système initial et Σ un système quelconque tel que $\Sigma_0 \Longrightarrow^* \Sigma$. (En particulier,* $\mathrm{var}(\Sigma_0) \subseteq \mathrm{var}(\Sigma)$ *d'après le lemme 4.10.)*

Alors

– *les règles et les équations de Σ sont correctes par rapport à Φ_0, et*

– *pour tout $\theta, \lambda \models^\sharp \Sigma$, on a $\theta, \lambda_0 \models^\sharp \Sigma_0$ où $\lambda_0 = \theta \uplus \lambda|_{\mathrm{var}^1(\Sigma_0)}$.*

En particulier, par le lemme précédent, $\theta, \lambda \models \Sigma$ implique $\theta, \lambda_0 \models \Sigma_0$ pour un certain $\lambda_0 =_E \theta \uplus \lambda|_{\mathrm{var}^1(\Sigma_0)}$.

Démonstration. Nous prouvons les propriétés voulues par récurrence sur la longueur de la dérivation $\Sigma_0 \Longrightarrow^* \Sigma$.

Le cas de base $\Sigma = \Sigma_0$ est clair par le point (3) du lemme précédent (avec $\Psi \mapsto \emptyset$, $\theta \mapsto \lambda|_{\mathcal{Y}}$, $\lambda \mapsto \lambda$, $\theta' \mapsto \theta$ et $\lambda' \mapsto \lambda_0$). Supposons que $\Sigma_0 \Longrightarrow^* \Sigma \Longrightarrow \Sigma'$, que les règles et les équations de Σ sont Φ_0-correctes, et que $\theta, \lambda' \models^\sharp \Sigma'$.

Nous traitons tout d'abord le cas des règles n'ajoutant pas de règles ni d'équations au système, à savoir **Project**, **Imitate**, **Coalesce**, **Narrowing** et **Constrain**. Pour toutes ces règles, on vérifie que $\theta, \lambda' \models^\sharp \Sigma'$ implique $\theta, \lambda \models^\sharp \Sigma$ pour $\lambda = \lambda'|_{\mathrm{var}(\Sigma)}$; d'où l'on déduit facilement les propriétés escomptées sur Σ' à partir de Σ. Nous considérons en détail le cas de **Project** et de **Narrowing**, les autres cas étant similaires ou plus simples.

Project. Avec les notations de la règle, supposons

$$\Sigma = \Phi;\, \Psi;\, \mathcal{C} \uplus \{X \rhd^? f(t_1, \ldots, t_n)\};\, \mathcal{N};\, \sigma \\ \Longrightarrow (\Phi;\, \Psi;\, \mathcal{C};\, \mathcal{N};\, \sigma)\,\{X \mapsto M\}\mu = \Sigma'$$

avec en particulier $(M \rhd t) \in \Phi$ et $\mu = \mathrm{mgu}(t, f(t_1, \ldots, t_n))$.

Soit $\bar{\mu} = \{X \mapsto M\}\mu$. Notez que $\bar{\mu}$ est bien formée et que $\bar{\mu}|_{\mathcal{X}^1} = \mu$ et $\bar{\mu}|_{\mathcal{X}^2} = \{X \mapsto M\}$.

L'hypothèse $\theta, \lambda' \models^\sharp \Sigma'$ signifie :

(i) pour tout $Y \rhd^? t$ dans $\mathcal{C}\bar{\mu}$, $Y\lambda' \rhd^E_{\Phi\bar{\mu}\lambda'} t\lambda'$;

(ii) pour toute équation $t_1 =^?_E t_2$ dans $\mathcal{C}\bar{\mu}$, $t_1\lambda' =_E t_2\lambda'$;

(iii) pour toute diséquation $t_1 \neq^?_E t_2$ dans $\mathcal{C}\bar{\mu}$, $t_1\lambda' \neq_E t_2\lambda'$;

(iv) pour tout $Y \in \mathrm{var}(\Sigma')$, $Y\sigma\bar{\mu}\lambda' = Y\lambda'$;

(v) pour tout $y \in \mathrm{var}(\Sigma')$, $y\sigma\bar{\mu}\lambda' =_E y\lambda'$;

(vi) pour tout $Y \in \mathcal{Y}$, $Y\theta \bowtie_{\Psi\bar{\mu}\lambda'} Y\lambda'$.

Sachant que $\Sigma\sigma = \Sigma$, en utilisant (iv) et (v), ceci implique :

(i) pour tout $Y \rhd^? t$ dans $\mathcal{C}$, $Y\lambda' \rhd^E_{\Phi\lambda'} t\lambda'$;

(ii) pour toute équation $t_1 =^?_E t_2$ dans $\mathcal{C}$, $t_1\lambda' =_E t_2\lambda'$;

(iii) pour toute diséquation $t_1 \neq^?_E t_2$ dans $\mathcal{C}$, $t_1\lambda' \neq_E t_2\lambda'$;

(iv) pour tout $Y \in \mathrm{var}(\Sigma')$, $Y\sigma\lambda' = Y\lambda'$;

(v) pour tout $y \in \mathrm{var}(\Sigma')$, $y\sigma\lambda' =_E y\lambda'$;

(vi) pour tout $Y \in \mathcal{Y}$, $Y\theta \bowtie_{\Psi\lambda'} Y\lambda'$.

Pour montrer $\theta, \lambda \models^\sharp \Sigma$ avec $\lambda = \lambda'|_{\mathrm{var}(\Sigma)}$, il reste à vérifier la contrainte $X \rhd^? f(t_1, \ldots, t_n)$. Or, $X\lambda' = M\lambda' \rhd_{\Phi\lambda'} t\lambda' =_E f(t_1 \ldots t_n)\lambda'$.

Enfin, sachant que $\theta, \lambda' \models^\sharp \Sigma$,

– pour tout $M \rhd t$ dans $\Phi\lambda'(= \Phi\lambda)$, on a $M[\Phi_0\lambda'] = M[\Phi_0\lambda] =_E t$; et

– pour tout $\forall\beta.M \bowtie N$ dans $\Psi\lambda'(= \Psi\lambda)$, on a $M[\Phi_0\lambda'] =_E N[\Phi_0\lambda']$.

Narrowing. Avec les notations de la règle, la transition s'écrit :

$$\Sigma = \Phi;\, \Psi;\, \mathcal{C};\, \mathcal{N};\, \sigma \\ \Longrightarrow \Sigma' = (\Phi\{t_0 \mapsto r\};\, \Psi;\, \mathcal{C}\{t_0 \mapsto r\};\, \mathcal{N} \cup \{t_1, \ldots, t_n\};\, \sigma)\mu$$

où $t_0 = f(t_1, \ldots, t_n)$, $\mu = \mathrm{mgu}(t_0, l)$ et $\mathrm{var}(\Sigma) \subseteq \mathrm{var}(\Sigma') \subseteq \mathrm{var}(\Sigma) \uplus \mathrm{var}(l)$.

Supposons $\theta, \lambda' \models^\sharp \Sigma'$:

(i) pour tout $Y \rhd^? t$ dans $\mathcal{C}\{t_0 \mapsto r\}\mu$, $Y\lambda' \rhd^E_{\Phi\{t_0\mapsto r\}\mu\lambda'} t\lambda'$;

(ii) pour toute équation $t_1 =^?_E t_2$ dans $\mathcal{C}\{t_0 \mapsto r\}\mu$, $t_1\lambda' =_E t_2\lambda'$;

(iii) pour toute diséquation $t_1 \neq^?_E t_2$ dans $\mathcal{C}\{t_0 \mapsto r\}\mu$, $t_1\lambda' \neq_E t_2\lambda'$;

(iv) pour tout $Y \in \mathrm{var}(\Sigma')$, $Y\sigma\mu\lambda' = Y\lambda'$;
(v) pour tout $y \in \mathrm{var}(\Sigma')$, $y\sigma\mu\lambda' =_E y\lambda'$;
(vi) pour tout $Y \in \mathcal{Y}$, $Y\theta \bowtie_{\Psi\mu\lambda'} Y\lambda'$.

Comme $t_0\mu\lambda' =_E r\mu\lambda'$ et $\Sigma\sigma = \Sigma$, en utilisant (iv) et (v), on obtient

(i) pour tout $Y \rhd^? t$ dans $\mathcal{C}$, $Y\lambda' \rhd^E_{\Phi\lambda'} t\lambda'$;
(ii) pour toute équation $t_1 =^?_E t_2$ dans $\mathcal{C}$, $t_1\lambda' =_E t_2\lambda'$;
(iii) pour toute diséquation $t_1 \neq^?_E t_2$ dans $\mathcal{C}$, $t_1\lambda' \neq_E t_2\lambda'$;
(iv) pour tout $Y \in \mathrm{var}(\Sigma')$, $Y\sigma\lambda' = Y\lambda'$;
(v) pour tout $y \in \mathrm{var}(\Sigma')$, $y\sigma\lambda' =_E y\lambda'$;
(vi) pour tout $Y \in \mathcal{Y}$, $Y\theta \bowtie_{\Psi\lambda'} Y\lambda'$.

si bien que $\theta, \lambda|_{\mathrm{var}(\Sigma)} \models^\sharp \Sigma$. La correction des règles et équations est identique au cas précédent.

Nous étudions maintenant la correction des quatre dernières règles : **Relate**, **Discard**, **Context-1** et **Context-2**. La correction de ces règles ne peut reposer sur la correction locale de $\models^\sharp$ (dans le sens $\theta \models^\sharp \Sigma' \Rightarrow \theta \models^\sharp \Sigma$) du fait des équations ajoutées au système. Nous utilisons à la place le lemme 4.11 et la correction des règles et des équations de Σ par rapport à Φ_0.

Relate. La transition s'écrit

$$\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \Longrightarrow \Phi;\ \Psi \cup \{X_0 \bowtie M_0\};\ \mathcal{C} \cup \{X_0 \rhd^? t_0\};\ \mathcal{N} \cup \{t_0\};\ \sigma = \Sigma'$$

où $(M_0 \rhd t_0) \in \Phi$ et X_0 est une variable fraîche d'arité k_0.

Supposons $\theta, \lambda' \models^\sharp \Sigma'$. Soit $\lambda = \lambda'|_{\mathrm{var}(\Sigma)}$. Les contraintes de Σ étant incluses dans celles de Σ' pour un ensemble de règles identique, on a $\lambda|_{\mathcal{Y}}, \lambda \models^\sharp \Sigma$.

Par hypothèse de récurrence, ceci implique $\lambda|_{\mathcal{Y}}, \lambda_1 \models^\sharp \Sigma_0$ pour $\lambda_1 = \lambda|_{\mathcal{Y}} \uplus \lambda|_{\mathrm{var}^1(\Sigma_0)}$. De plus, la correction des règles et des équations de Σ s'écrit :

– pour tout $M \rhd t$ dans $\Phi\lambda = \Phi\lambda'$, $M[\Phi_0\lambda_1] =_E t$;
– pour tout $\forall\beta.M \bowtie N$ dans $\Psi\lambda = \Psi\lambda'$, $M[\Phi_0\lambda_1] =_E N[\Phi_0\lambda_1]$.

Sachant que $(M_0 \rhd t_0) \in \Phi$ et que $X_0\lambda' \rhd^E_{\Phi\lambda'} t_0\lambda'$ du fait de $\theta, \lambda' \models^\sharp \Sigma'$, on en déduit

$$(X_0\lambda')[\Phi_0\lambda_1] =_E t_0\lambda' =_E (M_0\lambda')[\Phi_0\lambda_1].$$

Ainsi, les règles et les équations de Σ' sont Φ_0-correctes en ce qui concerne la solution $\lambda|_{\mathcal{Y}}, \lambda' \models^\sharp \Sigma'$.

Sachant que $\theta \bowtie_{\Psi\lambda' \cup \{X_0 \bowtie M_0\}\lambda'} \lambda|_{\mathcal{Y}}$ et $\lambda|_{\mathcal{Y}}, \lambda_1 \models^\sharp \Sigma_0$, le lemme 4.11 implique donc $\theta, \lambda_0 \models^\sharp \Sigma_0$ pour $\lambda_0 = \theta \uplus \lambda_1|_{\mathrm{var}^1(\Sigma_0)} = \theta \uplus \lambda'|_{\mathrm{var}^1(\Sigma_0)}$.

Discard. La preuve est identique au cas de **Relate**.

Context-1. Avec les notations de la règle, la transition s'écrit :

$$\begin{aligned}\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \Longrightarrow (\Phi \cup \{M' \rhd r\};\ \Psi \cup \{\forall Z_1, \ldots, Z_q.M \bowtie M'\};\\ \mathcal{C} \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\};\ \mathcal{N} \cup \{l_1, \ldots, l_n\};\ \sigma)\mu\\ = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma'\end{aligned}$$

Supposons $\theta, \lambda' \models^\sharp \Sigma'$. Soit $\lambda = \lambda'|_{\mathrm{var}(\Sigma)}$. Montrons pour commencer que $\lambda|_{\mathcal{Y}}, \lambda \models^\sharp \Sigma$.

En effet, comme $\Sigma\sigma = \Sigma$, $\mathrm{var}(\Sigma') \supseteq \mathrm{var}(\Sigma)$ et $\theta, \lambda' \models^\sharp \Sigma'$, on a d'une part

$$(\mu\lambda')|_{\mathrm{var}(\Sigma')} = (\sigma\mu\lambda')|_{\mathrm{var}(\Sigma')} = (\sigma'\lambda')|_{\mathrm{var}(\Sigma')} = \lambda'$$

et d'autre part pour tout $1 \leq j \leq p$, $Y_j\lambda' \rhd^E_{\Phi'\lambda'} y_j\lambda'$. Pour tout j, il existe donc un contexte public C_j et des règles $N_1 \rhd s_1, \ldots, N_m \rhd s_m$ dans Φ tel que

$$\begin{array}{rcl} Y_j\lambda' & = & C_j[N_1\lambda', \ldots, N_m\lambda', M'\lambda'] \\ y_j\lambda' & =_E & C_j[s_1\lambda', \ldots, s_m\lambda', r\lambda'] \end{array}$$

Comme $Y_j \in \mathrm{var}(M')$ et M' n'est pas une variable (D est propre), $M'\lambda'$ ne peut être un sous-terme de $Y_j\lambda'$, donc $w_{m+1} \notin \mathrm{par}(C_j)$. Par conséquent,

$$\begin{array}{rcl} Y_j\lambda' & = & C_j[N_1\lambda', \ldots, N_m\lambda'] \\ y_j\lambda' & =_E & C_j[s_1\lambda', \ldots, s_m\lambda'] \end{array}$$

autrement dit, $Y_j\lambda' \rhd^E_{\Phi\lambda'} y_j\lambda'$. Comme $\Phi\lambda' = \Phi\lambda$, $\mathrm{var}(r) \subseteq \mathrm{var}(l_1, \ldots, l_n)$, et pour tout $1 \leq i \leq n$, $t_i\lambda' = l_i\lambda'$, on en déduit

$$\begin{array}{rcl} M'\lambda' & = & D[M_1\lambda', \ldots, M_n\lambda', Y_1\lambda', \ldots, Y_p\lambda', \mathsf{c}_{\mathrm{sort}(Z_1)}, \ldots, \mathsf{c}_{\mathrm{sort}(Z_q)}] \\ & \rhd_{\Phi\lambda} & D[l_1\lambda', \ldots, l_n\lambda', y_1\lambda', \ldots, y_p\lambda', \mathsf{c}_{\mathrm{sort}(Z_1)}, \ldots, \mathsf{c}_{\mathrm{sort}(Z_q)}] \\ & = & l\{z_j \mapsto \mathsf{c}_{\mathrm{sort}(Z_j)}\}\lambda' \\ & =_E & r\{z_j \mapsto \mathsf{c}_{\mathrm{sort}(Z_j)}\}\lambda' \\ & = & r\lambda' \end{array}$$

L'équation ajoutée $M'\lambda' \rhd r\lambda'$ étant redondante (modulo E) des équations de $\Phi\lambda' = \Phi\lambda$, et vu la forme de Σ', $\lambda|_{\mathcal{Y}}, \lambda' \models^\sharp \Sigma'$ implique bien $\lambda|_{\mathcal{Y}}, \lambda \models^\sharp \Sigma$.

Par hypothèse de récurrence, ceci implique $\lambda|_{\mathcal{Y}}, \lambda_1 \models^\sharp \Sigma_0$ pour $\lambda_1 = \lambda|_{\mathcal{Y}} \uplus \lambda|_{\mathrm{var}^1(\Sigma_0)}$. De plus, la correction des règles et des équations de Σ s'écrit :

- pour tout $N \rhd t$ dans $\Phi\lambda = \Phi\lambda'$, $N[\Phi_0\lambda_1] =_E t$;
- pour tout $\forall\beta.N_1 \bowtie N_2$ dans $\Psi\lambda = \Psi\lambda'$, $N_1[\Phi_0\lambda_1] =_E N_2[\Phi_0\lambda_1]$.

En particulier, $M'\lambda' \rhd^E_{\Phi\lambda} r\lambda'$ implique $(M'\lambda')[\Phi_0\lambda_1] =_E r\lambda'$.

De plus, si $\lambda'' = \lambda'\|_{\mathrm{dom}(\lambda')\cup\{Z_1,\ldots,Z_q\}}$, alors par un calcul identique à ci-dessus

$$(M\lambda'')[\Phi_0\lambda_1] =_E r\lambda' =_E (M'\lambda'')[\Phi_0\lambda_1]$$

Ainsi, les règles et les équations de Σ' sont Φ_0-correctes en ce qui concerne la solution $\lambda|_{\mathcal{Y}}, \lambda' \models \Sigma'$.

Par le lemme 4.11, $\theta \bowtie_{\Psi'\lambda'} \lambda|_{\mathcal{Y}}$ et $\lambda|_{\mathcal{Y}}, \lambda_1 \models^\sharp \Sigma_0$ impliquent donc $\theta, \lambda_0 \models^\sharp \Sigma_0$ pour $\lambda_0 = \theta \uplus \lambda_1|_{\mathrm{var}^1(\Sigma_0)} = \theta \uplus \lambda'|_{\mathrm{var}^1(\Sigma_0)}$.

Context-2. La preuve est similaire au cas de **Context-1**. □

Pour conclure cette section, nous vérifions la préservation des systèmes standard, propriété attendue pour le bon déroulement global de la procédure (voir sous-section 4.1.2).

Proposition 4.13 (Systèmes standard)**.** *Soit Σ et Σ' deux systèmes tels que $\Sigma \Longrightarrow \Sigma'$. Si Σ est standard alors Σ' l'est également.*

Démonstration. Rappelons qu'un système $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ est standard (définition 4.4) lorsque l'une des conditions suivantes est vérifiées :

(a) $\mathcal{C}$ est de la forme $\mathcal{C} = \mathcal{C}_0 \uplus \{X^* \rhd^? x,\ Y^* \rhd^? y,\ x =^?_E y\}$ où $X^*, Y^*, x, y \notin \mathrm{var}(\Phi;\ \Psi;\ \mathcal{C}_0;\ \mathcal{N};\ \sigma)$;

(b) $\mathcal{C}$ et σ sont de la forme $\mathcal{C} = \mathcal{C}_0 \uplus \{X^* \rhd^? x,\ Y^* \rhd^? x\}$ et $\sigma = \sigma_0\{y \mapsto x\}$ où $X^*, Y^*, x, y \notin \mathrm{var}(\Phi;\ \Psi;\ \mathcal{C}_0;\ \mathcal{N};\ \sigma_0)$;

(c) $\mathcal{C}$ et σ sont de la forme $\mathcal{C} = \mathcal{C}_0 \uplus \{X^* \rhd^? x\}$, et $\sigma = \sigma_0\{y \mapsto x,\ Y^* \mapsto X^*\}$ où $X^*, Y^*, x, y \notin \mathrm{var}(\Phi;\ \Psi;\ \mathcal{C}_0;\ \mathcal{N};\ \sigma_0)$, quitte à inverser les rôles de X^* et de Y^*.

Soit $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ un système standard et supposons $\Sigma \Longrightarrow \Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma'$.

Étant données les conditions sur les occurrences des variables X^*, Y^*, x, y, par analyse des règles, les seuls cas où la transformation porte sur ces variables sont les suivants :

- la règle **Constrain** est appliquée sur Σ vérifiant (a) pour donner Σ' vérifiant (b) ;
- la règle **Coalesce** est appliquée sur Σ vérifiant (b) pour donner Σ' vérifiant (c).

Dans les autres cas, Σ' vérifie la même condition (a), (b) ou (c) que Σ. □

4.3 Résultats préliminaires à la complétude et à la terminaison

Définition 4.11. Un système de contraintes Σ est *accessible* ss'il existe un système initial Σ_0 tel que $\Sigma_0 \Longrightarrow^* \Sigma$.

Étant un ensemble de contraintes $\mathcal{C}$, soit de plus

$$\begin{aligned} \mathcal{C}_{\rhd} &\triangleq \{X \rhd^? t \in \mathcal{C}\}, \\ \mathcal{C}_{=} &\triangleq \{t =^?_E t' \in \mathcal{C}\} \quad \text{et} \\ \mathcal{C}_{\neq} &\triangleq \{t \neq^?_E t' \in \mathcal{C}\}. \end{aligned}$$

4.3.1 Étude des règles de pré-résolution

Nous montrons plusieurs lemmes techniques concernant les règles de pré-résolution (table 4.1). Il s'agit de contrôler notamment l'application de ces règles en vue d'établir différents invariants et de préparer l'étude des dérivations standard.

Étant donné un ensemble de contraintes $\mathcal{C}$, on note $\mathrm{nps}(\mathcal{C})$ l'ensemble des *termes du premier ordre non pré-résolus de* $\mathcal{C}$, défini de la manière suivante :

$$\mathrm{nps}(\mathcal{C}) \triangleq \{t \mid \exists (X \rhd^? t) \in \mathcal{C} \text{ tel que } t \notin \mathcal{X} \text{ ou bien } \exists (X' \rhd^? t) \in \mathcal{C}, X' \neq X\}$$

Par extension, $\mathrm{nps}(\Sigma)$ désigne l'ensemble $\mathrm{nps}(\mathcal{C})$ lorsque $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$.

Notons que Σ est pré-résolu ssi $\mathrm{nps}(\Sigma) = \emptyset$. De plus, l'opérateur nps est monotone dans le sens où $\mathcal{C} \subseteq \mathcal{C}' \Rightarrow \mathrm{nps}(\mathcal{C}) \subseteq \mathrm{nps}(\mathcal{C}')$.

Lemme 4.14. *Soit $\mathcal{C}$ un ensemble de contraintes et μ une substitution (bien formée) à support dans les variables du premier ordre :* $\mathrm{supp}(\mu) \subseteq \mathcal{X}^1$. *On a*

$$\text{pour tout } t, \qquad t\mu \in \mathrm{nps}(\mathcal{C}\mu) \text{ et } t \notin \mathrm{nps}(\mathcal{C}) \Rightarrow t \in \mathrm{var}(\mu)$$

En particulier, $\mathrm{nps}(\mathcal{C}\mu) \subseteq \mathrm{nps}(\mathcal{C})\mu \cup \mathrm{supp}(\mu)\mu$.

Démonstration. Soit $(X \rhd^? t) \in \mathcal{C}$ tel que $t\mu$ un élément de $\mathrm{nps}(\mathcal{C}\mu)$ et $t \notin \mathrm{nps}(\mathcal{C})$. En particulier, $t = x$ est une variable.

Si $x \in \mathrm{supp}(\mu)$, la propriété est claire. Sinon $x\mu = x$. Dans ce cas, comme $x\mu = x \in \mathrm{nps}(\mathcal{C}\mu) - \mathrm{nps}(\mathcal{C})$, il existe $(Y \rhd^? y) \in \mathcal{C}$ tel que $y \neq x$, $Y \neq X$ et $y\mu = x$; d'où $x\mu = x = y\mu \in \mathrm{supp}(\mu)\mu$.

Le deuxième point découle du fait que $\mathrm{var}(\mu)\mu \subseteq \mathrm{supp}(\mu)\mu$. □

Lemme 4.15. *Supposons* $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \Longrightarrow_{\mathbf{R}} \Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma'$ *pour une règle* **R** *de pré-résolution (***Coalesce***,* **Project** *ou* **Imitate***). Alors on a les faits suivants :*

(i) $\mathrm{nps}(\Sigma') \subseteq \mathrm{st}(\mathrm{nps}(\Sigma))\,\sigma'$.

(ii) Supposons que pour tout $(M \rhd t) \in \Phi$, $t \notin \mathcal{X}$. *Si* $\mathrm{nps}(\Sigma) = \{t_0\}$ *et si le terme (non variable)* t_0 *apparaît dans une seule contrainte de* $\mathcal{C}$, *alors* $\mathrm{nps}(\Sigma') \subseteq \mathrm{st}_{\neq}(t_0)\,\sigma'$.

Démonstration.

(i) Dans le cas de la règle **Coalesce**, la transition s'écrit :

$$\begin{aligned}\Sigma = \Phi;\ \Psi;\ \mathcal{C}_0 \cup \{X_1 \rhd^? t,\ X_2 \rhd^? t\};\ \mathcal{N};\ \sigma \\ \Longrightarrow \Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma' \\ = (\Phi;\ \Psi;\ \mathcal{C}_0 \cup \{X_1 \rhd^? t\};\ \mathcal{N};\ \sigma)\{X_2 \mapsto X_1\}\end{aligned}$$

En particulier, $\mathrm{nps}(\Sigma') \subseteq \mathrm{nps}(\Sigma) = \mathrm{nps}(\Sigma)\sigma'$.

Dans le cas de la règle **Imitate**, la transition s'écrit :

$$\begin{aligned}\Sigma = \Phi;\ \Psi;\ \mathcal{C}_0 \cup \{X \rhd^? f(t_1, \ldots, t_n)\};\ \mathcal{N};\ \sigma \\ \Longrightarrow \Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma' \\ = (\Phi;\ \Psi;\ \mathcal{C}_0 \cup \{X_1 \rhd^? t_1, \ldots, X_n \rhd^? t_n\};\ \mathcal{N};\ \sigma)\{X \mapsto f(X_1, \ldots, X_n)\}\end{aligned}$$

En particulier, on a $\mathrm{nps}(\Sigma') \subseteq \mathrm{nps}(\mathcal{C}_0) \cup \{t_1, \ldots, t_n\} \subseteq \mathrm{st}(\mathrm{nps}(\Sigma)) = \mathrm{st}(\mathrm{nps}(\Sigma))\sigma'$.

Considérons une instance de la règle **Project** de la forme :

$$\begin{aligned}\Sigma = \Phi;\ \Psi;\ \mathcal{C}_0 \cup \{X \rhd^? f(t_1, \ldots, t_n)\};\ \mathcal{N};\ \sigma \\ \Longrightarrow \Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma' \\ = (\Phi;\ \Psi;\ \mathcal{C}_0;\ \mathcal{N};\ \sigma)\{X \mapsto M\}\mu\end{aligned}$$

Par le lemme 4.14, on a $\mathrm{nps}(\Sigma') = \mathrm{nps}(\mathcal{C}_0\mu) \subseteq \mathrm{nps}(\mathcal{C}_0)\mu \cup \mathrm{supp}(\mu)\mu$.

Or, d'une part, $\mathrm{nps}(\mathcal{C}_0)\mu \subseteq \mathrm{nps}(\Sigma)\mu \subseteq \mathrm{st}(\mathrm{nps}(\Sigma))\mu = \mathrm{st}(\mathrm{nps}(\Sigma))\sigma'$.

D'autre part, par définition $\mathrm{supp}(\mu) \subseteq \mathrm{var}(t, f(t_1, \ldots, t_n))$ et $t\mu = f(t_1, \ldots, t_n)\mu$. Pour tout $y \in \mathrm{supp}(\mu)$, on a donc $y\mu \in \mathrm{var}(f(t_1, \ldots, t_n)\mu)$. Par conséquent, $\mathrm{supp}(\mu)\mu \subseteq \mathrm{st}(f(t_1, \ldots, t_n)\mu) \subseteq \mathrm{st}(\mathrm{nps}(\Sigma))\sigma'$.

(ii) Si $\mathrm{nps}(\Sigma) = \{t_0\}$, où le terme (non variable) t_0 apparaît dans une seule contrainte de $\mathcal{C}$, alors les seules règles applicables sont **Project** et **Imitate** sur le t_0 en question. Le cas de **Imitate** est clair. Pour **Project**, on note que dans la preuve du point précédent, on a $\mathrm{nps}(\mathcal{C}_0) = \emptyset$, donc $\mathrm{nps}(\Sigma') \subseteq \mathrm{supp}(\mu)\mu$. Or, $\mathrm{supp}(\mu) \subseteq \mathrm{var}(t, f(t_1, \ldots, t_n)) \subseteq \mathrm{st}_{\neq}(t, f(t_1, \ldots, t_n))$ car par hypothèse t n'est pas une variable. □

Grâce au lemme précédent, nous montrons deux invariants syntaxiques simples des systèmes accessibles.

Proposition 4.16. *Pour tout système accessible* $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$, $\mathrm{nps}(\Sigma) \subseteq \mathrm{st}(\mathcal{N})$ *et pour tout* $x \in \mathrm{supp}(\sigma)$, $x\sigma \in \mathrm{st}(\mathcal{N})$.

Notamment, comme conséquence du premier invariant $\mathrm{nps}(\Sigma) \subseteq \mathrm{st}(\mathcal{N})$, le remplacement $\{t_0 \mapsto r\}$ de la règle **Narrowing** ne concerne jamais les contraintes de déductibilité d'un système accessible $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ dans la mesure où :

$$\{t \mid (X \rhd^? t) \in \mathcal{C}\} \subseteq \mathrm{nps}(\Sigma) \cup \mathcal{X}^1 \subseteq \mathrm{st}(\mathcal{N}) \cup \mathcal{X}$$

Démonstration. Si Σ est initial, les deux propriétés sont immédiates. Supposons $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \Longrightarrow \Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma'$ où $\mathrm{nps}(\Sigma) \subseteq \mathrm{st}(\mathcal{N})$ et pour tout $x \in \mathrm{supp}(\sigma)$, $x\sigma \in \mathrm{st}(\mathcal{N})$.

D'après le lemme 4.15, dans le cas des règles **Project**, **Imitate** et **Coalesce**, on a

$$\mathrm{nps}(\Sigma') \subseteq \mathrm{st}(\mathrm{nps}(\Sigma))\sigma' \subseteq \mathrm{st}(\mathcal{N})\sigma' \subseteq \mathrm{st}(\mathcal{N}').$$

De plus, pour **Imitate** et **Coalesce**, $\sigma'|_{\mathcal{X}^1} = \sigma|_{\mathcal{X}^1}$. Concernant **Project**, on a comme précédemment $\mathrm{supp}(\mu)\mu \subseteq \mathrm{st}(f(t_1, \ldots, t_n)\mu) \subseteq \mathrm{nps}(\Sigma)\sigma' \subseteq \mathrm{st}(\mathcal{N})\sigma'$. Sachant que $\sigma' = \sigma\mu$ avec $\mathrm{supp}(\sigma) \cap \mathrm{var}(\mu) = \emptyset$, σ et μ sont idempotentes et $\mathrm{supp}(\sigma)\sigma \subseteq \mathrm{st}(\mathcal{N})$, on a donc bien

$$\begin{aligned} \mathrm{supp}(\sigma')\sigma' &= \mathrm{supp}(\sigma)\sigma' \cup \mathrm{supp}(\mu)\sigma' \\ &= \mathrm{supp}(\sigma)\sigma\sigma' \cup \mathrm{supp}(\mu)\mu\sigma' \\ &\subseteq \mathrm{st}(\mathcal{N})\sigma' \end{aligned}$$

Le cas des règles **Relate** et **Discard** est clair.

Concernant **Narrowing**, la règle s'écrit :

$$\begin{aligned} \Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \Longrightarrow \Sigma' &= \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma' \\ &= (\Phi\{t_0 \mapsto r\};\ \Psi;\ \mathcal{C}\{t_0 \mapsto r\};\ \mathcal{N} \cup \{t_1, \ldots, t_n\};\ \sigma)\mu \end{aligned}$$

où $l \to r$ est une règle issue de $\mathcal{R}$, $t_0 = f(t_1, \ldots, t_n)$ et $\mu = \mathrm{mgu}(t_0, l)$. D'après l'hypothèse de récurrence sur Σ et la remarque faite plus haut, t_0 n'apparaît pas dans les contraintes de déductibilité de Σ. Ainsi on a $\mathcal{C}_{\rhd}\{t_0 \mapsto r\} = \mathcal{C}_{\rhd}$. Par le lemme 4.14, on en déduit

$$\mathrm{nps}(\Sigma') = \mathrm{nps}(\mathcal{C}\{t_0 \mapsto r\}\mu) \subseteq \mathrm{nps}(\mathcal{C})\mu \cup \mathrm{supp}(\mu)\mu.$$

Or, d'une part $\mathrm{nps}(\mathcal{C})\mu \subseteq \mathrm{st}(\mathcal{N})\mu \subseteq \mathrm{st}(\mathcal{N}')$. D'autre part, $\mathrm{supp}(\mu) \subseteq \mathrm{var}(l, t_0) \subseteq \mathrm{st}_{\neq}(l, t_0)$ car l et t_0 ne sont pas des variables ($\mathcal{R}$ étant terminant). Ainsi, $\mathrm{supp}(\mu)\mu \subseteq \mathrm{st}_{\neq}(t_0\mu) \subseteq \mathrm{st}(\mathcal{N}')$; d'où $\mathrm{nps}(\Sigma') \subseteq \mathrm{st}(\mathcal{N}')$. Par ailleurs, $\mathrm{supp}(\mu)\mu \subseteq \mathrm{st}(\mathcal{N}')$ implique $\mathrm{supp}(\sigma')\sigma' \subseteq \mathrm{st}(\mathcal{N}')$ comme ci-dessus.

Les règles **Constrain** et **Context-{1,2}** se traitent de manière similaire. □

4.3.2 Pré-ordre de stratification

Nous établissons plusieurs lemmes techniques concernant la relation de stratification $\preceq$, qui joue un rôle important dans la suite de preuve. Rappelons que la notation $M_1 \preceq M_2$ signifie que
– $\mathrm{maxpar}(M_1) \leq \mathrm{maxpar}(M_2)$, et
– pour tout $Y \in \mathrm{var}(M_1)$, ou bien $Y \in \mathrm{var}(M_2)$ ou bien $\mathrm{ar}(Y) < \mathrm{maxpar}(M_2)$.
Cette notion bénéficie des propriétés suivantes.

Lemme 4.17.

(1) $\preceq$ *est un pré-ordre (partiel), c'est-à-dire une relation réflexive et transitive ;*

(2) $\preceq$ *est stable par application de contextes : pour tout* C *tel que* $C[M_1, \ldots, M_n]$ *existe, si pour tout* $1 \leq i \leq n$, $M_i \preceq M_i'$ *et* $\mathrm{sort}(M_i) = \mathrm{sort}(M_i')$, *alors*

$$C[M_1, \ldots, M_n] \preceq C[M_1', \ldots, M_n']$$

(3) $\preceq$ *est stable par application de substitutions (bien formées) : pour tout* λ,

$$M_1 \preceq M_2 \;\; \textit{implique} \;\; M_1\lambda \preceq M_2\lambda$$

(4) pour tout M, X, *on a*
  (i) $X \preceq M$ *ssi* $X \in \mathrm{var}(M)$ *ou* $\mathrm{ar}(X) < \mathrm{maxpar}(M)$,
  (ii) $M \preceq X$ *ssi* $\mathrm{par}(M) = \emptyset$ *et* $\mathrm{var}(M) \subseteq \{X\}$;

(5) les deux conditions suivantes sont équivalentes :
  (i) $M_1 \preceq M_2$ *et* $M_2 \notin \mathrm{var}(M_1)$,
  (ii) pour toute substitution (bien formée) λ, $\mathrm{maxpar}(M_1\lambda) \leq \mathrm{maxpar}(M_2\lambda)$, *et* $M_2\lambda \notin \mathrm{st}(\mathrm{var}(M_1)\lambda)$.

Démonstration.

(1) $\preceq$ est clairement réflexive. Concernant la transitivité, supposons $M_1 \preceq M_2 \preceq M_3$. On a $\mathrm{maxpar}(M_1) \leq \mathrm{maxpar}(M_3)$. Soit $Y \in \mathrm{var}(M_1)$. On a ou bien $\mathrm{ar}(Y) < \mathrm{maxpar}(M_2) \leq \mathrm{maxpar}(M_3)$, ou bien $Y \in \mathrm{var}(M_2)$. Dans le dernier cas, par hypothèse, $Y \in \mathrm{var}(M_3)$ ou $\mathrm{ar}(Y) < \mathrm{maxpar}(M_3)$.

(2) Soit C tel que $C[M_1, \ldots, M_n]$ soit un terme valide, et supposons que pour tout $1 \leq i \leq n$, $M_i \preceq M_i'$. En particulier, $\mathrm{maxpar}(M_i) \leq \mathrm{maxpar}(M_i')$ d'où $\mathrm{maxpar}(C[M_1, \ldots, M_n]) \preceq \mathrm{maxpar}(C[M_1', \ldots, M_n'])$.
Soit $Y \in \mathrm{var}(C[M_1, \ldots, M_n])$. Comme C est clos (par définition), il existe i tel que $w_i \in \mathrm{par}(M)$ et $Y \in \mathrm{var}(M_i)$. Par hypothèse, ou bien

$$Y \in \mathrm{var}(M_i') \subseteq \mathrm{var}(C[M_1', \ldots, M_n']),$$

ou bien $\mathrm{ar}(Y) < \mathrm{maxpar}(M_i') \leq \mathrm{maxpar}(C[M_1', \ldots, M_n'])$.

(3) Supposons $M_1 \preceq M_2$ et soit λ une substitution bien formée.
Soit $w_i \in \mathrm{par}(M_1\lambda)$. (a) Si $w_i \in \mathrm{par}(M_1)$, alors $i \leq \mathrm{maxpar}(M_1) \leq \mathrm{maxpar}(M_2) \leq \mathrm{maxpar}(M_2\lambda)$. (b) Sinon, il existe $Y \in \mathrm{var}(M_1)$ tel que $w_i \in \mathrm{par}(Y\lambda)$. Par hypothèse, ou bien $Y \in \mathrm{var}(M_2)$ ou bien $\mathrm{ar}(Y) < \mathrm{maxpar}(M_2)$. Dans le premier cas, on déduit $w_i \in \mathrm{par}(M_2\lambda)$, et $i \leq \mathrm{maxpar}(M_2\lambda)$. Dans le second cas, comme λ est bien formée, on a $i \leq \mathrm{ar}(Y) < \mathrm{maxpar}(M_2) \leq \mathrm{maxpar}(M_2\lambda)$.

Soit $Y \in \mathrm{var}(M_1\lambda)$. Il existe $X \in \mathrm{var}(M_1)$ tel que $Y \in \mathrm{var}(X\lambda)$. Par hypothèse, on a donc $X \in \mathrm{var}(M_2)$ ou $\mathrm{ar}(X) < \mathrm{maxpar}(M_2)$. Dans le premier cas, on a $Y \in \mathrm{var}(M_2\lambda)$. Dans le second cas, comme λ est bien formée, on a $\mathrm{ar}(Y) \leq \mathrm{ar}(X) < \mathrm{maxpar}(M_2) \leq \mathrm{maxpar}(M_2\lambda)$.

(4) Les deux propriétés découlent directement des définitions.

(5) Supposons que $M_1 \preceq M_2$, et M_2 n'est pas une variable de M_1. Soit λ une substitution bien formée. Par la propriété (3), on a $\mathrm{maxpar}(M_1\lambda) \leq \mathrm{maxpar}(M_2\lambda)$. Supposons qu'il existe une variable $X \in \mathrm{var}(M_1)$ telle que $M_2\lambda$ est un sous-terme de $X\lambda$. Alors, par hypothèse, ou bien $X \in \mathrm{var}(M_2)$ ou bien $\mathrm{ar}(X) < \mathrm{maxpar}(M_2)$. Comme $M_2\lambda$ est un sous-terme de $X\lambda$, le premier cas implique $X = M_2$, donc $M_2 \in \mathrm{var}(M_1)$; contradiction. Dans le second cas, comme λ est bien formée, $\mathrm{maxpar}(X\lambda) \leq \mathrm{ar}(X) < \mathrm{maxpar}(M_2) \leq \mathrm{maxpar}(M_2\lambda)$ ce qui contredit le fait que $M_2\lambda$ est un sous-terme de $X\lambda$.

Réciproquement, supposons que pour tout λ, $\mathrm{maxpar}(M_1\lambda) \leq \mathrm{maxpar}(M_2\lambda)$ et $M_2\lambda \notin \mathrm{st}(\mathrm{var}(M_1)\lambda)$. Premièrement, en prenant $\lambda = \{\}$, on obtient que M_2 n'est pas une variable de M_1, et $\mathrm{maxpar}(M_1) \leq \mathrm{maxpar}(M_2)$. De plus, pour tout $Y \in \mathrm{var}(M_1)$, on a $Y \in \mathrm{var}(M_2)$ ou $\mathrm{ar}(Y) < \mathrm{maxpar}(M_2)$. En effet, supposons qu'il existe $Y \in \mathrm{var}(M_1)$ tel que $Y \notin \mathrm{var}(M_2)$ et $\mathrm{ar}(Y) \geq \mathrm{maxpar}(M_2)$. La substitution bien formée $\lambda = \{Y \mapsto M_2\}$ est telle que $M_2\lambda = M_2 = Y\lambda \in \mathrm{st}(\mathrm{var}(M_1)\lambda)$. □

La définition de $\preceq$ est motivée par le lemme suivant, qui est utilisée de manière récurrente dans la preuve de complétude pour transformer la solution d'un système en une solution d'un système légèrement modifié.

Lemme 4.18. *Soit $\mathcal{U}_0$ un système d'équations de la forme*

$$\begin{cases} X_1 & =^? & M_1 \\ & \cdots & \\ X_m & =^? & M_m \end{cases}$$

où $X_1, \ldots, X_m$ sont m variables du second ordre distinctes et $M_1, \ldots, M_m$ sont des termes arbitraires du second ordre. On appelle solution *d'un tel système toute substitution partielle close (bien formée) λ telle que* $\mathrm{dom}(\lambda) \supseteq \mathrm{var}(\mathcal{U}_0)$ *et* $\forall i, X_i\lambda_i = M_i\lambda$.

Soit $X_0 \in \mathrm{var}(M_1, \ldots, M_m) - \{X_1, \ldots, X_m\}$ une variable et M et N deux termes.

Soit $\mathcal{U} = \mathcal{U}_0\{X_0 \mapsto M\}$ et $\mathcal{U}' = \mathcal{U}_0\{X_0 \mapsto N\}$ les systèmes obtenus en remplaçant dans $\mathcal{U}_0$ la variable X_0 par M et N, respectivement.

Supposons que

(1) aucun membre droit d'équation dans $\mathcal{U}$ n'est une variable :

$$\forall i, \quad M_i\{X_0 \mapsto M\} \notin \mathcal{X},$$

(2) il existe une solution λ de $\mathcal{U}$ telle que $\mathrm{dom}(\lambda) \supseteq \mathrm{var}(N)$, $\mathrm{maxpar}(N\lambda) \leq \mathrm{maxpar}(M\lambda)$ *et* $M\lambda \notin \mathrm{st}(\mathrm{var}(N)\lambda)$.

Alors il existe une solution λ' de $\mathcal{U}'$ telle que

(a) $\mathrm{dom}(\lambda') = \mathrm{dom}(\lambda)$,

(b) $M\lambda' = M\lambda$,

(c) $N\lambda' = N\lambda$,

(d) pour tout $V \in \mathrm{dom}(\lambda) - \{X_1, \ldots, X_m\}$, $V\lambda' = V\lambda$, *et*

(e) pour tout $1 \leq i \leq m$,
- $X_i\lambda' \bowtie_{\{M\lambda' \bowtie N\lambda'\}} X_i\lambda$,
- $\mathrm{maxpar}(X_i\lambda') \leq \mathrm{maxpar}(X_i\lambda)$,
- *et dans le cas où* $X_i\lambda' \neq X_i\lambda$, $M\lambda$ *est un sous-terme de* $X_i\lambda$.

Enfin, par la propriété (5) du lemme 4.17, si $N \preceq M$ *et* $M \notin \mathrm{var}(N)$, *alors la condition (2) ci-dessus est vraie pour* toute *solution* λ *de* $\mathcal{U}$ *telle que* $\mathrm{dom}(\lambda) \supseteq \mathrm{var}(N)$.

Démonstration. Notons $i \prec' j$ lorsque $X_i \in \mathrm{var}(M_j\{X_0 \mapsto N\})$. On montre tout d'abord que $\prec'$ n'a pas de cycle $i_1 \prec' \ldots \prec' i_k \prec' i_1$ pour $k \geq 1$.

En effet, si $X_i \in \mathrm{var}(M_j\{X_0 \mapsto N\})$, alors on a au choix :

(i) $X_i \in \mathrm{var}(M_j) - \{X_0\} \subseteq \mathrm{var}(M_j\{X_0 \mapsto M\})$, d'où on déduit par l'hypothèse (1) et par $X_j\lambda = M_j\{X_0 \mapsto M\}\lambda$ que $X_i\lambda$ est un sous-terme strict de $X_j\lambda$; ou bien

(ii) $X_i \in \mathrm{var}(N)$ et $X_0 \in \mathrm{var}(M_j)$: en particulier $M\lambda$ est un sous-terme de $X_j\lambda$.

Supposons qu'il existe un cycle $i_1 \prec' \ldots \prec' i_k \prec' i_1$ avec $k \geq 1$. Comme le point (i) ne peut s'appliquer à tous les indices successifs, il existe une sous-chaîne de taille minimale, entre des indices i et j, commençant et finissant par le cas (ii). En particulier, $M\lambda$ est un sous-terme de $X_i\lambda$, $X_i\lambda$ est sous-terme de $X_j\lambda$ et $X_j \in \mathrm{var}(N)$ ce qui contredit l'hypothèse (2).

Par conséquent, par induction sur $\prec'$, il existe un unique λ' tel que
- (a) $\mathrm{dom}(\lambda') = \mathrm{dom}(\lambda)$,
- (d) pour tout $V \in \mathrm{dom}(\lambda) - \{X_1, \ldots, X_m\}$, $V\lambda' = V\lambda$,
- pour tout $1 \leq i \leq m$, $X_i\lambda' = M_i\{X_0 \mapsto N\}\lambda'$.

Par induction sur $\prec'$, étant donné que $\mathrm{maxpar}(N\lambda) \leq \mathrm{maxpar}(M\lambda)$ (hypothèse (2)) et que $\mathrm{var}(N\lambda) = \emptyset$, on a bien $\mathrm{maxpar}(X_i\lambda') \leq \mathrm{maxpar}(X_i\lambda)$ et $\mathrm{var}(X_i\lambda') \subseteq \mathrm{var}(X_i\lambda)$; ainsi λ' est clos et bien formé. De plus, une récurrence similaire montre que pour tout i, $X_i\lambda' \bowtie_{M\lambda \bowtie N\lambda} X_i\lambda$.

Notons $i \prec j$ lorsque $X_i \in \mathrm{var}(M_j\{X_0 \mapsto M\})$. Par l'hypothèse (1), $i \prec j$ implique que $X_i\lambda$ est un sous-terme strict de $M_j\{X_0 \mapsto M\}\lambda = X_j\lambda$. En particulier, la relation $\prec$ n'a pas de cycle. Soit $\prec^*$ la clôture transitive de $\prec$.

Montrons la propriété suivante :

(f) pour tout $1 \leq j \leq m$, si $X_j\lambda' \neq X_j\lambda$ alors il existe $i \prec^* j$ tel que $X_0 \in \mathrm{var}(M_i)$ et $M\lambda$ est sous-terme de $X_j\lambda$, de plus strict si $i \neq j$.

En effet, soit $1 \leq j \leq m$. Si pour tout $i \prec^* j$, $X_0 \notin \mathrm{var}(M_i)$, alors par induction sur $\prec$, on en déduit $X_j\lambda' = X_j\lambda$. De plus, si $i \prec^* j$ et $X_0 \in \mathrm{var}(M_i)$, alors $M\lambda$ est sous-terme de $M_i\{X_0 \mapsto M\}\lambda = X_i\lambda$ lui-même sous-terme de $X_j\lambda$, strict si $i \neq j$ d'après ce qui précède.

Nous utilisons maintenant (f) pour montrer (b) $M\lambda' = M\lambda$, (c) $N\lambda' = N\lambda$, et conclure (e).

(b) Si $M\lambda' \neq M\lambda$, alors il existe $X_j \in \mathrm{var}(M)$ tel que $X_j\lambda' \neq X_j\lambda$. Par (f), ceci implique que $M\lambda$ est sous-terme de $M_i\{X_0 \mapsto M\}\lambda = X_i\lambda$ lui-même sous-terme de $X_j\lambda$ (strict si $i \neq j$). On en déduit $i = j$, $M_j = X_0$ et $M = X_j$; en particulier $M_j\{X_0 \mapsto M\} = X_j$ ce qui contredit l'hypothèse (1).

(c) Si $N\lambda' \neq N\lambda$, alors il existe $X_j \in \mathrm{var}(N)$ tel que $X_j\lambda' \neq X_j\lambda$. Par (f), ceci implique que $M\lambda$ est sous-terme de $X_j\lambda$, ce qui contredit l'hypothèse (2).

(e) On a vu que pour tout i, $\mathrm{maxpar}(X_i\lambda') \leq \mathrm{maxpar}(X_i\lambda)$. De plus, par les points (b) et (c), la relation $X_i\lambda' \bowtie_{M\lambda \bowtie N\lambda} X_i\lambda$ implique $X_i\lambda' \bowtie_{M\lambda' \bowtie N\lambda'} X_i\lambda$. Enfin, (f) montre en particulier que si $X_i\lambda' \neq X_i\lambda$, alors $M\lambda$ est un sous-terme de $X_i\lambda$. □

Ce lemme confirme l'intuition motivant la définition du pré-ordre de stratification (définition 4.6) : lorsque $N \preceq M$ (et que M n'est pas une variable), alors N est une recette plus générale que M.

4.3.3 Invariant de stratification

Nous établissons ici un invariant syntaxique important, correspondant à la condition de régularité (2) des systèmes initiaux (section 3.4).

Proposition 4.19 (Invariant de stratification)**.** *Soit $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}$; σ un système accessible. Alors Σ est* stratifié*, dans le sens où pour tout $(M \rhd t) \in \Phi$,*

- *M n'est pas une variable, et*
- *pour tout $x \in \mathrm{var}(t)$, il existe une contrainte $X \rhd^? t'$ dans $\mathcal{C}$ telle que $x \in \mathrm{var}(t')$ et $X \preceq M$, c'est-à-dire $X \in \mathrm{var}(M)$ ou $\mathrm{ar}(X) < \mathrm{maxpar}(M)$.*

En particulier, le second point implique $\mathrm{var}^1(\Phi) \subseteq \mathrm{var}^1(\mathcal{C}_\rhd)$.

Démonstration. La condition est satisfaite sur les systèmes de contraintes initiaux du fait de la condition de régularité (2) (section 3.4). Supposons que $\Sigma \Longrightarrow \Sigma'$ et que Σ soit stratifié. On vérifie que Σ' est stratifié en fonction de la règle utilisée.

- **Project**. Supposons

$$\Sigma = \Phi \cup \{M \rhd t\}; \Psi; \mathcal{C} \uplus \{X \rhd^? f(t_1, \ldots, t_n)\}; \mathcal{N}\,\sigma$$
$$\Longrightarrow \Sigma' = (\,\Phi \cup \{M \rhd t\}; \Psi; \mathcal{C}; \mathcal{N}\,\sigma\,)\{X \mapsto M\}\mu$$

avec les notations de la table 4.1. En particulier, $X \in \mathrm{var}(M)$, $\mathrm{ar}(X) \geq \mathrm{maxpar}(M)$, et $\mathrm{ar}(X) \geq \mathrm{maxar}(M)$. Soit $\rho = \{X \mapsto M\}$. μ et ρ sont bien formés et agissent respectivement sur les variables du premier et du second ordre. En particulier, $\rho\mu = \mu\rho$.
Soit $M_0 \rhd s_0$ une règle de $\Phi \cup \{M \rhd t\}$ et $x \in \mathrm{var}(s_0\mu)$. Nous vérifions qu'il existe $X_1 \rhd^? s_1$ dans $\mathcal{C}$ tel que $x \in \mathrm{var}(s_1\mu)$ et $X_1\rho \preceq M_0\rho$.
Soit $y \in \mathrm{var}(s_0)$ tel que $x \in \mathrm{var}(y\mu)$. Par la condition de stratification sur $M_0 \rhd s_0$ dans Σ, il existe $Y \rhd^? t'$ dans $\mathcal{C} \cup \{X \rhd^? f(t_1, \ldots, t_n)\}$ tel que $y \in \mathrm{var}(t')$ et $Y \preceq M_0$.
Si $(Y \rhd^? t') \neq (X \rhd^? f(t_1, \ldots, t_n))$, alors la contrainte $Y \rhd^? t'$ appartient à $\mathcal{C}$. On choisit alors $X_1 = Y$, $s_1 = t'$. En effet, par le lemme 4.17, $Y \preceq M_0$ implique $Y\rho \preceq M_0\rho$.
Dans le cas contraire, on a $(Y \rhd^? t') = (X \rhd^? f(t_1, \ldots, t_n))$ ce qui implique $y \in \mathrm{var}(f(t_1, \ldots, t_n))$ et $X \preceq M_0$. Comme $x \in \mathrm{var}(f(t_1, \ldots, t_n)\mu) = \mathrm{var}(t\mu)$ par définition de μ, il existe $y' \in \mathrm{var}(t)$ tel que $x \in \mathrm{var}(y'\mu)$.
Par stratification sur $M \rhd t$ dans Σ, il existe $Y' \rhd^? t''$ dans $\mathcal{C} \cup \{X \rhd^? f(t_1, \ldots, t_n)\}$ tel que $y' \in \mathrm{var}(t'')$ et $Y' \preceq M$. Comme $X \not\preceq M$, on a en particulier $Y' \neq X$, donc $Y' \rhd^? t'' \in \mathcal{C}$.

On choisit $X_1 = Y'$ et $s_1 = t''$, et l'on vérifie que $Y' \preceq M_0\rho$. En effet, sachant que $X \preceq M_0$ et $Y' \preceq M$, quatre cas peuvent se présenter :
- si $\mathrm{ar}(X) < \mathrm{maxpar}(M_0)$ et $X \notin \mathrm{var}(M_0)$,
  - si $Y' \in \mathrm{var}(M)$, alors par les prémisses de la règle,

$$\mathrm{ar}(Y') \leq \mathrm{maxar}(M) \leq \mathrm{ar}(X) < \mathrm{maxpar}(M_0\rho)$$

  - si $\mathrm{ar}(Y') < \mathrm{maxpar}(M)$, alors

$$\begin{array}{rcll} \mathrm{ar}(Y') & < & \mathrm{maxpar}(M) & \\ & \leq & \mathrm{ar}(X) & \text{(par les prémisses)} \\ & < & \mathrm{maxpar}(M_0\rho) & \end{array}$$

- si $X \in \mathrm{var}(M_0)$,
  - si $Y' \in \mathrm{var}(M)$, alors $Y'\rho \in \mathrm{var}(M)$ et comme $X \in \mathrm{var}(M_0)$, $Y' \in \mathrm{var}(M_0\rho)$;
  - enfin, si $\mathrm{ar}(Y') < \mathrm{maxpar}(M)$, alors de même

$$\mathrm{ar}(Y') < \mathrm{maxpar}(M) \leq \mathrm{maxpar}(M_0\rho).$$

- **Imitate** et **Coalesce**. La propriété découle des conditions sur les arités des variables.
- **Constrain** et **Context-2**. Notons que si Σ est stratifié, alors le système $\Sigma\mu$ également. En effet, soit $M_0 \rhd s_0$ dans Φ et $x \in \mathrm{var}(s_0\mu)$. Il existe une variable $y \in \mathrm{var}(s_0)$ tel que $x \in \mathrm{var}(y\mu)$. Par stratification de $M_0 \rhd s_0$ dans Σ, il existe $X_0 \rhd^? s_1$ dans $\mathcal{C}$ tel que $y \in \mathrm{var}(s_1)$ et $X_0 \preceq M_0$.
  La contrainte $X_0 \rhd^? s_1\mu$ de $\mathcal{C}\mu$ vérifie donc $x \in \mathrm{var}(s_1\mu)$ et $X_0 \preceq M_0$.
- **Narrowing**. Comme ci-dessus $\Sigma\mu = (\Phi;\, \Psi;\, \mathcal{C};\, \mathcal{N};\, \sigma)\mu$ est stratifié. Comme $\mathrm{var}(r) \subseteq \mathrm{var}(l)$, on a $\mathrm{var}(r\mu) \subseteq \mathrm{var}(l\mu) = \mathrm{var}(t_0\mu)$. Or, d'après la proposition 4.16, $t_0 \notin \mathrm{st}(\mathcal{N})$ n'est pas sous-terme de contraintes de déductibilité $X \rhd^? t$ dans $\mathcal{C}$. Par conséquent, le système

$$\Sigma' = (\Phi\{t_0 \mapsto r\};\, \Psi;\, \mathcal{C}_\rhd \cup (\mathcal{C}_= \cup \mathcal{C}_\neq)\{t_0 \mapsto r\};\, \mathcal{N} \cup \{t_1, \ldots, t_n\};\, \sigma)\mu$$

  est stratifié.
- **Context-1**. Ce cas se traite de la même manière que **Narrowing**, du fait que $\mathrm{var}(r) \subseteq \mathrm{var}(l_1, \ldots, l_m)$.
- Enfin, le cas des règles **Relate**, **Discard** est clair. □

La propriété de stratification est un ingrédient essentiel des preuves de complétude et de terminaison. Nous avons vu (fait 4.4) que la condition de régularité (2) impliquait notamment qu'une solution était uniquement déterminée par ses valeurs sur les variables du second ordre. Le lemme suivant généralise ce fait aux systèmes pré-résolus et stratifiés, justifiant l'intérêt de cette notion.

Lemme 4.20. *Soit $M_1 \rhd t_1$, …, $M_n \rhd t_n$ des règles d'environnement, et $X_1 \rhd^? x_1$, …, $X_m \rhd^? x_m$ des contraintes de déductibilité telles que les X_i et x_i sont deux à deux distinctes. Supposons que*

(i) pour tout j, M_j n'est pas une variable ;

(ii) pour tous i, j tels que $x_i \in \mathrm{var}(t_j)$, $X_i \preceq M_j$.

Alors, pour tous contextes publics $C_1, \ldots, C_m$, si le problème d'unification suivant $\mathcal{U}_2$:

$$\begin{cases} X_1 & =^? & C_1[M_1, \ldots, M_n] \\ & \cdots & \\ X_m & =^? & C_m[M_1, \ldots, M_n] \end{cases}$$

admet une solution bien formée alors, le problème correspondant sur les variables du premier ordre $\mathcal{U}_1$:

$$\begin{cases} x_1 & =^? & C_1[t_1, \ldots, t_n] \\ & \cdots & \\ x_m & =^? & C_m[t_1, \ldots, t_n] \end{cases}$$

est en forme DAG-résolue, c'est-à-dire que la relation $\prec$ définie par $i \prec j \Leftrightarrow x_i \in \mathrm{var}(C_j[t_1, \ldots, t_n])$ n'admet pas de cycle *: $i_1 \prec i_2 \ldots \prec i_k \prec i_1$ pour $k \geq 1$.*

Plus précisément, pour toute solution λ de $\mathcal{U}_2$ et pour tout $i \prec j$, on a

- $\mathrm{maxpar}(X_i\lambda) < \mathrm{maxpar}(X_j\lambda)$*, ou bien*
- $\mathrm{maxpar}(X_i\lambda) = \mathrm{maxpar}(X_j\lambda)$ *et* $X_i\lambda \in \mathrm{st}_{\neq}(X_j\lambda)$.

En particulier, si $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}$; σ est un système pré-résolu stratifié tel que

$$\begin{array}{rcl} \Phi & = & \{w_1 \rhd t_1, \ldots, w_{k_0} \rhd t_{k_0}\} \\ \mathcal{C}_{\rhd} & = & \{X_1 \rhd^? x_1, \ldots, X_m \rhd^? x_m\} \end{array}$$

alors les conditions (i) et (ii) sont remplies. Comme $\mathrm{var}(t_1, \ldots, t_n) \subseteq \{x_1, \ldots, x_m\}$*, $\mathcal{U}_1$ décrit ainsi les valeurs des x_i de manière univoque.*

Démonstration. Soit λ une solution de $\mathcal{U}_2$. Montrons que $i \prec j$ implique
- $\mathrm{maxpar}(X_i\lambda) < \mathrm{maxpar}(X_j\lambda)$, ou
- $\mathrm{maxpar}(X_i\lambda) = \mathrm{maxpar}(X_j\lambda)$ et $X_i \in \mathrm{st}_{\neq}(X_j\lambda)$.

Il s'ensuit que $\prec$ n'a pas de cycles.

En effet, supposons $x_i \in \mathrm{var}(C_j[t_1, \ldots, t_n])$. Sachant que C_j est clos (par définition), soit t_k tel que $x_i \in \mathrm{var}(t_k)$ et $w_k \in \mathrm{par}(C_j)$, autrement dit, t_k (respectivement M_k) est un sous-terme de $C_j[t_1, \ldots, t_n]$ (respectivement de $C_j[M_1, \ldots, M_n]$). Nous distinguons deux cas correspondants à l'hypothèse (ii) (pour $j = k$).

- Si $X_i \in \mathrm{var}(M_k)$, alors, comme par l'hypothèse (i) M_k n'est pas une variable, X_i est un sous-terme strict de M_k, donc de $C_j[M_1, \ldots, M_n]$. Ainsi $X_i\lambda$ est sous-terme strict de $C_j[M_1, \ldots, M_n]\lambda = X_j\lambda$ et

$$\mathrm{maxpar}(X_i\lambda) \leq \mathrm{maxpar}(C_j[M_1, \ldots, M_n]\lambda) = \mathrm{maxpar}(X_j\lambda).$$

- Si $\mathrm{ar}(X_i) < \mathrm{maxpar}(M_k)$, alors en utilisant le fait que λ est bien formé, on a

$$\begin{array}{rcl} \mathrm{maxpar}(X_i\lambda) & \leq & \mathrm{ar}(X_i) \\ & < & \mathrm{maxpar}(M_k) \\ & \leq & \mathrm{maxpar}(C_j[M_1, \ldots, M_n]) \\ & \leq & \mathrm{maxpar}(C_j[M_1, \ldots, M_n]\lambda) \\ & = & \mathrm{maxpar}(X_j\lambda) \end{array}$$

□

4.3.4 Évolution des relations $\bowtie_\Psi$ et $\rhd_{\Phi,\mathcal{C}}$

Étant donnés des ensembles Φ, $\mathcal{N}$ et $\mathcal{C}$, notons

$$\Phi^{\mathcal{N}} \triangleq \{M \rhd t \in \Phi \mid t \in \mathrm{st}(\mathcal{N}) \cup \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}}\}$$

l'ensemble des règles de Φ dont le membre droit est un sous-terme de $\mathcal{N}$ ou un terme clos en forme $\mathcal{R}$-normale.

Nous étudions maintenant l'évolution des relations $\bowtie_\Psi$, $\rhd_{\Phi,\mathcal{C}}$ et $\rhd_{\Phi^{\mathcal{N}},\mathcal{C}}$ le long d'une dérivation.

Lemme 4.21. *Soit $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ et $\Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma'$ deux systèmes de contraintes tels que $\Sigma \Longrightarrow_{\mathbf{R}} \Sigma'$ par une certaine règle* **R**. *Soit M et t deux termes.*

(1) Si **R** *est une des règles suivantes :* **Project**, **Imitate**, **Coalesce**, **Constrain**, **Context-{1,2}**, *ou* **Relate**, *alors*

$$\begin{aligned} M \rhd_{\Phi,\mathcal{C}} t &\Rightarrow M\sigma' \rhd_{\Phi',\mathcal{C}'} t\sigma' \\ \Sigma \text{ accessible et } M \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t &\Rightarrow M\sigma' \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t\sigma' \end{aligned}$$

(2) Si **R** *est la règle* **Narrowing**, *alors*

$$\begin{aligned} M \rhd_{\Phi,\mathcal{C}} t &\Rightarrow \exists t',\quad M\sigma' \rhd_{\Phi',\mathcal{C}'} t' \leftarrow^*_{\mathcal{R}} t\sigma' \\ \Sigma \text{ accessible et } M \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t &\Rightarrow M\sigma' \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t\sigma' \end{aligned}$$

(3) Si **R** *est la règle* **Discard**, *alors $\sigma' = \sigma$ et*

$$\begin{aligned} M \rhd_{\Phi,\mathcal{C}} t &\Rightarrow \exists M',\quad M\,(\succeq \cap \bowtie_{\Psi'})\,M' \rhd_{\Phi',\mathcal{C}'} t \\ \Sigma \text{ accessible et } M \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t &\Rightarrow \exists M',\quad M\,(\succeq \cap \bowtie_{\Psi'})\,M' \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t \end{aligned}$$

Démonstration. Supposons $\Sigma = \Phi_0;\ \Psi_0;\ \mathcal{C}_0;\ \mathcal{N}_0;\ \sigma_0 \Longrightarrow \Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma'$, et soit M_0 et t_0 deux termes. Nous considérons alternativement les deux hypothèses :

(a) $M_0 \rhd_{\Phi_0,\mathcal{C}_0} t_0$,

(b) $M_0 \rhd_{\Phi_0{}^{\mathcal{N}_0},\mathcal{C}_0} t_0$,

(1) En reprenant les notations de chaque règle, pour **Project**, si

$$\begin{aligned} M_0 &= C[M_1, \ldots, M_m, Y_1, \ldots, Y_p, X] \quad \text{et} \\ t_0 &= C[s_1, \ldots, s_m, y_1, \ldots, y_p, f(t_1, \ldots, t_n)] \end{aligned}$$

où $(M_i \rhd s_i) \in \Phi$ et $(Y_j \rhd^? y_j) \in \mathcal{C}$ alors

$$\begin{aligned} M_0\sigma' &= C[M_1\sigma', \ldots, M_m\sigma', Y_1, \ldots, Y_p, M\sigma'] \quad \text{et} \\ t_0\sigma' &= C[s_1\sigma', \ldots, s_m\sigma', y_1\sigma', \ldots, y_p\sigma', t\sigma'] \end{aligned}$$

avec $(M_i \rhd s_i)\sigma' \in \Phi'$ et $(Y_j \rhd^? y_j\sigma') \in \mathcal{C}'$.
Dans le cas (a), on a donc bien $M_0\sigma' \rhd_{\Phi',\mathcal{C}'} t_0\sigma'$. Dans le cas (b), la proposition 4.16 implique $f(t_1, \ldots, t_n) \in \mathrm{st}(\mathcal{N})$, d'où $t\sigma' = f(t_1, \ldots, t_n)\sigma' \in \mathrm{st}(\mathcal{N}')$ et par conséquent, si pour tout i, $(M_i \rhd s_i) \in \Phi^{\mathcal{N}}$ alors $M_0\sigma' \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t_0\sigma'$.

Pour **Imitate**, si

$$\begin{aligned} M_0 &= C[M_1, \ldots, M_m, Y_1, \ldots, Y_p, X] \quad \text{et} \\ t_0 &= C[s_1, \ldots, s_m, y_1, \ldots, y_p, f(t_1, \ldots, t_n)] \end{aligned}$$

où $(M_i \rhd s_i) \in \Phi$ et $(Y_j \rhd^? y_j) \in \mathcal{C}$ alors

$$\begin{aligned} M_0\sigma' &= C[M_1\sigma', \ldots, M_m\sigma', Y_1, \ldots, Y_p, f(X_1, \ldots, X_n)] \quad \text{et} \\ t_0\sigma' = t_0 &= C[s_1, \ldots, s_m, y_1, \ldots, y_p, f(t_1, \ldots, t_n)] \end{aligned}$$

ce qui permet de conclure dans les deux cas (a) et (b).
Pour **Coalesce**, on note que $\Phi' = \Phi\sigma'$, $\Psi' = \Psi\sigma'$, $\mathcal{C}' = \mathcal{C}\sigma'$, $\mathcal{N}' = \mathcal{N}$.
Les cas de **Constrain**, **Context-{1,2}**, et **Relate** sont clairs dans la mesure où il existe une substitution μ tel que $\sigma' = \sigma\mu$, $\mathrm{supp}(\mu) \subseteq \mathcal{X}^1$ et de plus $\Phi' \supseteq \Phi\sigma'$, $\Psi' \supseteq \Psi\sigma'$, $\mathcal{C}'_\rhd \supseteq \mathcal{C}_\rhd\sigma'$ et $\mathcal{N}' \supseteq \mathcal{N}\sigma'$.

(2) **Narrowing**. On procède comme ci-dessus sachant que dans le cas (a) $t\sigma' \to^*_{\mathcal{R}} t'$ correspond à la surréduction dans Φ et $\mathcal{C}_\rhd$. Dans le cas (b), la surréduction ne concerne pas $\Phi^{\mathcal{N}}$ (par définition de la règle) ni $\mathcal{C}_\rhd$ (par la proposition 4.16).

(3) **Discard**. Avec les notations de la règle si

$$\begin{aligned} M_0 &= C[M_1, \ldots, M_m, Y_1, \ldots, Y_p, M] \quad \text{et} \\ t_0 &= C[s_1, \ldots, s_m, y_1, \ldots, y_p, t] \end{aligned}$$

avec $(M_i \rhd s_i) \in \Phi$ et $(Y_j \rhd^? y_j) \in \mathcal{C}$ alors sachant que $\bowtie$ et $\preceq$ sont stables par application de contextes (respectivement par définition et par le lemme 4.17),

$$\begin{aligned} M_0\sigma' = M_0 \quad \left\{ \begin{array}{c} \bowtie_{\Psi'} \\ \succeq \end{array} \right. \quad & C[M_1, \ldots, M_m, Y_1, \ldots, Y_p, N] \quad \text{et} \\ t_0\sigma' = t_0 \quad = \quad & C[s_1, \ldots, s_m, y_1, \ldots, y_p, t] \end{aligned}$$

avec par construction $N \rhd_{\Phi,\mathcal{C}} t$ ce qui prouve la propriété dans le cas (a). Pour le cas (b), si $M \rhd t$ apparaît dans la décomposition ci-dessus, c'est que par hypothèse $t \in \mathrm{st}(\mathcal{N}) \cup \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}}$. Dans ces conditions, noter que $N \rhd_{\Phi,\mathcal{C}} t$ implique $N \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t$. □

Ce lemme nous permet d'établir une propriété de progression (syntaxique) du système de règles : intuitivement si un terme est atteignable à un instant donné par application d'un contexte public sur Φ et $\mathcal{C}$, il reste atteignable plus tard par un calcul plus petit pour l'ordre de stratification $\preceq$ et relié au calcul initial par les équations découvertes.

Proposition 4.22. *Soit $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ et $\Sigma' = \Phi'; \Psi'; \mathcal{C}'; \mathcal{N}'; \sigma'$ deux systèmes de contraintes tels que $\Sigma \Longrightarrow^* \Sigma'$. Alors*

(1) pour tous M et t tels que $M \rhd_{\Phi,\mathcal{C}} t$, il existe M' et t' tel que

$$M\sigma' \,(\succeq \cap \bowtie_{\Psi'})\, M' \rhd_{\Phi',\mathcal{C}'} t' \leftarrow^*_{\mathcal{R}} t\sigma'$$

(2) supposons Σ accessible ; alors pour tous M et t tels que $M \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t$, il existe M' tel que

$$M\sigma' \,(\succeq \cap \bowtie_{\Psi'})\, M' \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t\sigma'$$

Démonstration. Dans les deux cas, on procède par induction sur la longueur de la dérivation, en utilisant le lemme précédent, le lemme 4.10 et le fait que les relations $\bowtie_\Psi$ et $\succeq$ sont transitives et stables par substitution (respectivement par définition et par le lemme 4.17). □

4.3.5 Étude des dérivations standard

Avant d'entamer les preuves de complétude et de terminaison, nous étudions plus particulièrement les dérivations standard. (Les paragraphes précédents portaient sur des stratégies d'application quelconques des règles.)

Application des règles de pré-résolution

Lors d'une dérivation standard, les règles principales **Narrowing**, **Constrain**, **Context-{1,2}** et **Relate** sont suivies par l'application des règles de pré-résolution. Nous étudions ici certaines propriétés de ces séquences de règles dans le prolongement de la sous-section 4.3.1.

Pour commencer, notons que l'on peut généraliser le pré-ordre de stratification $\preceq$ à des ensembles finis de termes du second ordre A et B : $A \preceq B$ ssi

- $\mathrm{maxpar}(A) \leq \mathrm{maxpar}(B)$ et
- pour tout $X \in \mathrm{var}(A)$, ou bien $X \in \mathrm{var}(B)$ ou bien $\mathrm{ar}(X) < \mathrm{maxpar}(B)$.

Cette généralisation est cohérente avec la définition initiale (définition 4.6) si l'on voit chaque ensemble $A = \{M_1, \ldots, M_n\}$ comme un terme $M_A = f(M_1, \ldots, M_n)$ pour un certain symbole f. On en déduit par le lemme 4.17 que la notation généralisée est encore un pré-ordre (partiel) stable par substitution (bien formée).

Étant donné un ensemble de contraintes $\mathcal{C}$, notons $\mathrm{vnps}(\mathcal{C})$ l'ensemble défini par

$$\mathrm{vnps}(\mathcal{C}) \triangleq \{X \mid \exists t \in \mathrm{nps}(\mathcal{C}) \cup \mathrm{var}(\mathrm{nps}(\mathcal{C})),\ (X \rhd^? t) \in \mathcal{C}\}$$

et par extension, posons $\mathrm{vnps}(\Sigma) \triangleq \mathrm{vnps}(\mathcal{C})$ lorsque $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$.

Intuitivement, il s'agit d'un sur-ensemble des variables du second ordre pouvant être substituées par une règle de pré-résolution dans Σ.

Lemme 4.23. *Soit $\mathcal{C}$ un ensemble de contraintes et μ une substitution (bien formée) de support inclus dans $\mathcal{X}^1$. On a*

$$\mathrm{vnps}(\mathcal{C}\mu) \subseteq \mathrm{vnps}(\mathcal{C}) \cup \{Y \mid \exists y \in \mathrm{var}(\mu),\ (Y \rhd^? y) \in \mathcal{C}\}$$

Démonstration. Soit $Y \in \mathrm{vnps}(\mathcal{C}\mu)$: il existe $(Y \rhd^? s) \in \mathcal{C}$ tel que que $s\mu$ vérifie $s\mu \in \mathrm{nps}(\mathcal{C}\mu)$ ou $s\mu \in \mathrm{var}(\mathrm{nps}(\mathcal{C}\mu))$. Si $s \in \mathrm{nps}(\mathcal{C}) \cup \mathrm{var}(\mathrm{nps}(\mathcal{C}))$, alors $Y \in \mathrm{vnps}(\mathcal{C})$. Sinon, s est une certaine variable $y = s \notin \mathrm{var}(\mathrm{nps}(\mathcal{C}))$.

Si $y \in \mathrm{supp}(\mu)$, alors la propriété est vérifiée. Sinon, on a $y\mu = y \in \mathrm{var}(\mathrm{nps}(\mathcal{C}\mu))$. Autrement dit, il existe une contrainte $(Z \rhd^? t) \in \mathcal{C}$ tel que $t\mu \in \mathrm{nps}(\mathcal{C}\mu)$, $y \in \mathrm{var}(t\mu)$ sachant par ailleurs que $y \notin \mathrm{var}(\mathrm{nps}(\mathcal{C}))$.

Si $t \in \mathrm{nps}(\mathcal{C})$, alors $y \notin \mathrm{var}(\mathrm{nps}(\mathcal{C}))$ implique en particulier $y \notin \mathrm{var}(t)$. Comme $y \in \mathrm{var}(t\mu)$, on en déduit $y \in \mathrm{var}(\mathrm{supp}(\mu)\mu)$.

Sinon, $t \notin \mathrm{nps}(\mathcal{C})$ et $t\mu \in \mathrm{nps}(\mathcal{C}\mu)$ impliquent par lemme 4.14 que $t \in \mathrm{var}(\mu)$, d'où $y \in \mathrm{var}(t\mu) \subseteq \mathrm{var}(\mathrm{supp}(\mu)\mu)$. □

Lemme 4.24. *Supposons* $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \Longrightarrow_{\mathbf{R}} \Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma'$ *pour une règle* **R** *de pré-résolution — en l'occurrence* **Coalesce**, **Project** *ou* **Imitate**. *Supposons de plus* Σ *accessible (ou simplement stratifié au sens de la proposition 4.19). Alors,*

$$\mathrm{vnps}(\Sigma') \preceq \mathrm{vnps}(\Sigma)\sigma'$$

Démonstration. Dans le cas de la règle **Coalesce**, la transition s'écrit :

$$\begin{aligned}\Sigma = \Phi;\ \Psi;\ \mathcal{C}_0 \cup \{X_1 \rhd^? t,\ X_2 \rhd^? t\};\ \mathcal{N};\ \sigma \\ \Longrightarrow \Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma' \\ = (\Phi;\ \Psi;\ \mathcal{C}_0 \cup \{X_1 \rhd^? t\};\ \mathcal{N};\ \sigma)\{X_2 \mapsto X_1\}\end{aligned}$$

En particulier, $\mathrm{vnps}(\Sigma') \subseteq \mathrm{vnps}(\mathcal{C}_0 \cup \{X_1 \rhd^? t\}) = \mathrm{vnps}(\Sigma) - \{X_2\} = \mathrm{vnps}(\Sigma)\sigma'$.

Dans le cas de la règle **Imitate**, la transition s'écrit :

$$\begin{aligned}\Sigma = \Phi;\ \Psi;\ \mathcal{C}_0 \cup \{X \rhd^? f(t_1, \ldots, t_n)\};\ \mathcal{N};\ \sigma \\ \Longrightarrow \Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma' \\ = (\Phi;\ \Psi;\ \mathcal{C}_0 \cup \{X_1 \rhd^? t_1, \ldots, X_n \rhd^? t_n\};\ \mathcal{N};\ \sigma)\{X \mapsto f(X_1, \ldots, X_n)\}\end{aligned}$$

D'où on déduit $\mathrm{vnps}(\Sigma') \subseteq \mathrm{vnps}(\Sigma) \cup \{X_1, \ldots, X_n\} = \mathrm{vnps}(\Sigma)\sigma'$.

Considérons la règle **Project**. Avec les notations de la règle, on a

$$\begin{aligned}\Sigma = \Phi;\ \Psi;\ \mathcal{C}_0 \cup \{X \rhd^? f(t_1, \ldots, t_n)\};\ \mathcal{N};\ \sigma \\ \Longrightarrow \Sigma' = \Phi';\ \Psi';\ \mathcal{C}';\ \mathcal{N}';\ \sigma' \\ = (\Phi;\ \Psi;\ \mathcal{C}_0;\ \mathcal{N};\ \sigma)(X \mapsto M)\mu\end{aligned}$$

Comme $X \notin \mathrm{var}(\mathcal{C}_0)$ et par le lemme 4.23, on a

$$\mathrm{vnps}(\Sigma') = \mathrm{vnps}(\mathcal{C}_0\mu) \subseteq \mathrm{vnps}(\mathcal{C}_0) \cup \{Y \mid \exists y \in \mathrm{var}(\mu), (Y \rhd^? y) \in \mathcal{C}_0\}$$

D'une part $\mathrm{vnps}(\mathcal{C}_0) = \mathrm{vnps}(\mathcal{C}_0)\sigma' \subseteq \mathrm{vnps}(\Sigma)\sigma'$. D'autre part, soit $Y \rhd^? y$ dans $\mathcal{C}_0$ tel que $y \in \mathrm{var}(\mu)$. Or par construction de μ, on a $\mathrm{var}(\mu) \subseteq \mathrm{var}(t, f(t_1, \ldots, t_n))$.

- Si $y \in \mathrm{var}(f(t_1, \ldots, t_n))$, alors $y \in \mathrm{nps}(\mathcal{C})$ et $Y \in \mathrm{vnps}(\mathcal{C}) \subseteq \mathrm{vnps}(\Sigma)\sigma'$.
- Sinon, $y \in \mathrm{var}(t)$. Par stratification de Σ, sachant que $M \rhd t$ est dans Φ, il existe $(Z \rhd^? s) \in \mathcal{C}$ tel que $y \in \mathrm{var}(s)$ et $Z \preceq M$.
  Si s n'est pas une variable ou bien si $s = y$ et $Y \neq Z$ alors $y \in \mathrm{var}(\mathrm{nps}(\mathcal{C}))$, d'où $Y \in \mathrm{vnps}(\mathcal{C}) \subseteq \mathrm{vnps}(\Sigma)\sigma'$.
  Sinon, on a $Y = Z \preceq M = X\sigma'$. Enfin, comme $f(t_1, \ldots, t_n) \in \mathrm{nps}(\Sigma)$, on a $X \in \mathrm{vnps}(\Sigma)$. Donc $Y \preceq \mathrm{vnps}(\Sigma)\sigma'$. □

Des lemmes 4.15 et 4.24 on déduit les propriétés suivantes utiles pour la preuve de complétude. Il s'agit ici de contrôler l'application des règles de pré-résolution à la suite d'une règle principale **Relate**, **Context-{1,2}** ou **Narrowing**.

Corollaire 4.25. *Soit* $\Sigma_0 = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ *un système pré-résolu. Supposons que pour tout* $(M \rhd t) \in \Phi$, t *n'est pas une variable.*

Supposons de plus qu'il existe une règle $(M_0 \rhd t_0) \in \Phi$ *tel que*

$$\Sigma_0 \Longrightarrow_{\mathbf{Relate}} \Sigma = \Phi;\ \Psi \cup \{X \bowtie M_0\};\ \mathcal{C} \cup \{X \rhd^? t_0\};\ \mathcal{N} \cup \{t_0\};\ \sigma$$

et que l'on ait $\Sigma \Longrightarrow^+ \Sigma'$ *par une dérivation non vide utilisant uniquement les règles* **Coalesce**, **Project** *et* **Imitate**. *On a alors*

$$\mathrm{nps}(\Sigma') \subseteq \mathrm{st}_{\neq}(t_0\sigma').$$

En particulier, pour tous $t \in \mathrm{nps}(\Sigma')$ *et* $\theta, \lambda' \models \Sigma'$, *alors* $\sigma'\lambda' = \lambda'$ *implique* $t\lambda' \in \mathrm{st}_{\neq}(t_0\lambda')$.

Démonstration. Sachant que pour tout $(M \rhd t) \in \Phi$, $t \notin \mathcal{X}$ (en particulier $t_0 \notin \mathcal{X}$) et que Σ_0 est pré-résolu, d'après le point (ii) du lemme 4.15, tout Σ_1 tel que $\Sigma \Longrightarrow \Sigma_1$ par une règle de pré-résolution vérifie

$$\mathrm{nps}(\Sigma_1) \subseteq \mathrm{st}_{\neq}(t_0\sigma_1)$$

où σ_1 est la dernière composante de Σ_1. Par induction et d'après le point (i) du lemme 4.15, tout Σ' tel que $\Sigma' \Longrightarrow \Sigma_1 \Longrightarrow^* \Sigma'$ vérifie donc

$$\mathrm{nps}(\Sigma') \subseteq \mathrm{st}(\mathrm{nps}(\Sigma_1)\sigma') \subseteq \mathrm{st}(\mathrm{st}_{\neq}(t_0\sigma_1)\sigma') \subseteq \mathrm{st}_{\neq}(t_0\sigma') \qquad \square$$

Corollaire 4.26. *Soit* $\Sigma_0 = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ *un système pré-résolu accessible. Supposons* $\Sigma_0 \Longrightarrow_{\mathbf{R}} \Sigma$ *pour une règle* $\mathbf{R}$ *parmi* **Context-{1,2}** *:* Σ *est de la forme*

$$\Sigma = (\Phi''; \Psi \cup \{\forall Z_1, \ldots, Z_q.M \bowtie M'\}; \\ \mathcal{C} \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\}; \mathcal{N} \cup \{l_1, \ldots, l_n\}; \sigma)\mu$$

avec les notations de la règles et en posant $\Phi'' = \Phi \cup \{M' \rhd r\}$ *dans le cas* $\mathbf{R} = \mathbf{Context-1}$ *et* $\Phi'' = \Phi$ *dans le cas contraire.*

Supposons de plus $\Sigma \Longrightarrow^* \Sigma' = \Phi'; \Psi'; \mathcal{C}'; \mathcal{N}'; \sigma'$ *par une dérivation utilisant uniquement les règles* **Coalesce**, **Project** *et* **Imitate**. *On a alors*

$$\begin{array}{rcll} \mathrm{nps}(\Sigma') & \subseteq & \mathrm{st}(l_1\sigma', \ldots, l_n\sigma') & \quad et \\ \mathrm{vnps}(\Sigma') & \preceq & M_0\sigma' & \end{array}$$

où $M_0 = M\{Z_j \mapsto \mathsf{c}_{\mathrm{sort}(Z_j)}\} = D[M_1, \ldots, M_n, Y_1, \ldots, Y_p, \mathsf{c}_{\mathrm{sort}(Z_1)}, \ldots, \mathsf{c}_{\mathrm{sort}(Z_q)}]$.

En particulier, dans le cas de **Context-1**, $M_0 = M'$ *est le membre gauche de la règle* $M' \rhd r\mu$ *ajoutée à* Σ_0. *Soit* k' *un entier tel que* $k' \geq \mathrm{maxpar}(M')$ *et* $k' \geq \mathrm{maxar}(M')$, *et soit*

$$\Phi_1 \stackrel{\Delta}{=} \{(M \rhd t) \in \Phi \mid \mathrm{maxpar}(M) \leq k'\}.$$

Pour toute contrainte $(X \rhd^? t) \in \mathcal{C}'$, *si* $t \in \mathrm{nps}(\Sigma')$ *et si* $\theta, \lambda' \models \Sigma'$ *alors*

$$X\lambda' \rhd_{(\Phi_1\sigma')\lambda'} t\lambda'$$

où $\Phi_1\sigma' \subseteq \Phi'$.

Démonstration. Vérifions les deux propriétés pour le système Σ : $\mathrm{nps}(\Sigma) \subseteq \mathrm{st}(l_1\mu, \ldots, l_n\mu)$ et $\mathrm{vnps}(\Sigma) \preceq M_0\sigma\mu = M_0$. Le résultat général découle ensuite des lemmes 4.15 et 4.24, par récurrence sur la longueur de la dérivation ($\preceq$ étant un pré-ordre stable par substitution).

Σ_0 étant pré-résolu et par fraîcheur des $y_1, \ldots, y_p$, on a

$$\begin{aligned} \mathrm{nps}(\mathcal{C} \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\}) &= \emptyset \\ \mathrm{vnps}(\mathcal{C} \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\}) &= \emptyset \end{aligned}$$

Par le lemme 4.14, on en déduit

$$\mathrm{nps}(\Sigma) \subseteq \mathrm{supp}(\mu)\mu \subseteq \mathrm{st}(l_1\mu, \ldots, l_n\mu) = \mathrm{st}(l_1\sigma', \ldots, l_n\sigma').$$

D'autre part, soit $Y \in \mathrm{vnps}(\Sigma)$ quelconque. Par le lemme 4.23, pour la même raison que ci-dessus, il existe $y \in \mathrm{var}(\mu)$ tel que $(Y \rhd^? y) \in \mathcal{C}_0 \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\}$). Si $Y \in \{Y_1, \ldots, Y_p\}$, alors $Y \preceq M_0$.

Sinon, comme $\mathrm{var}(\mu) \subseteq \mathrm{var}(t_1, \ldots, t_n, l_1, \ldots, l_n)$ et $\mathrm{var}(l_1, \ldots, l_n) \cap \mathrm{var}(\Sigma) = \emptyset$, on a $y \in \mathrm{var}(t_1, \ldots, t_n)$. Supposons $y \in \mathrm{var}(t_i)$. Σ étant stratifié (proposition 4.19), il existe $X \rhd^? t$ dans $\mathcal{C}$ tel que $X \preceq M_i$ et $y \in \mathrm{var}(t)$. Mais Σ est pré-résolu, donc $t = y$ et finalement $Y = X \preceq M_i \preceq M_0$.

Concernant le dernier point, du fait que $\theta, \lambda' \models \Sigma'$, on a

$$X\lambda' \rhd_{\Phi'\lambda'} t\lambda'$$

Toute règle de Φ' intervenant dans cette relation s'écrit $(M \rhd t')\sigma'$ où $(M \rhd t') \in \Phi \cup \{M' \rhd r\}$ et où $(M\sigma')\lambda' = M\lambda'$ est un sous-terme de $X\lambda'$.

Or, d'une part, $\mathrm{maxpar}(M) \leq \mathrm{ar}(X)$. Comme $t \in \mathrm{nps}(\Sigma')$ et par conséquent $X \in \mathrm{vnps}(\Sigma')$, on a $X \preceq M'\sigma'$, c'est-à-dire $\mathrm{ar}(X) < \mathrm{maxpar}(M'\sigma')$ ou $X \in \mathrm{var}(M'\sigma')$. Comme σ' est bien formée, on obtient $\mathrm{ar}(X) \leq \mathrm{maxpar}(M')$ ou $\mathrm{ar}(X) \leq \mathrm{maxar}(M')$; soit dans les deux cas, $\mathrm{maxpar}(M) \leq \mathrm{ar}(X) \leq k'$.

D'autre part, si $(M \rhd t') = (M' \rhd r)$, alors $(M\sigma')\lambda' = M'\lambda'$ est un sous-terme de $X\lambda'$ ce qui par le lemme 4.17 contredit le fait que $X \preceq M'\sigma'$ et $M' \neq X$ (par stratification de Σ').

Par conséquent, $(M \rhd t') \in \Phi_1$. Finalement, on a montré $X\lambda' \rhd_{(\Phi_1\sigma')\lambda'} t\lambda'$. □

Corollaire 4.27. *Soit $\Sigma_0 = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ un système pré-résolu accessible. Supposons $\Sigma_0 \Longrightarrow_{\mathbf{Narrowing}} \Sigma$. Avec les notations de la règle, Σ s'écrit*

$$\Sigma = (\Phi\{t_0 \mapsto r\}; \Psi; \mathcal{C}\{t_0 \mapsto r\}; \mathcal{N} \cup \{t_1, \ldots, t_n\}; \sigma)\mu$$

où $t_0 = f(t_1, \ldots, t_n)$ et $\mu = \mathrm{mgu}(t_0, l)$.

Supposons de plus $\Sigma \Longrightarrow^* \Sigma' = \Phi'; \Psi'; \mathcal{C}'; \mathcal{N}'; \sigma'$ par une dérivation utilisant uniquement les règles **Coalesce**, **Project** *et* **Imitate**. *On a alors*

$$\mathrm{nps}(\Sigma') \subseteq \mathrm{st}(t_1\sigma', \ldots, t_n\sigma').$$

De plus, notons $\Phi_0 = \{(M_0 \rhd t_0') \in \Phi \mid t_0 \in \mathrm{st}(t_0')\}$. Pour tout $(M_0 \rhd t_0') \in \Phi_0$, on a

$$\mathrm{vnps}(\Sigma') \preceq M_0\sigma'.$$

En particulier, soit

$$k_1 = \min\{\max(\mathrm{maxpar}(M_0), \mathrm{maxar}(M_0)) \mid (M_0 \rhd t_0') \in \Phi_0\} \in [0, \infty]$$

et

$$\Phi_1 = \{(M \rhd t) \in \Phi - \Phi_0 \mid \mathrm{maxpar}(M) \leq k_1\} \subseteq \Phi\{t_0 \mapsto r\}$$

Pour toute contrainte $(X \rhd^? t) \in \mathcal{C}'$*, si* $t \in \mathrm{nps}(\Sigma')$ *et si* $\theta, \lambda' \models \Sigma'$ *alors*

$$X\lambda' \rhd_{(\Phi_1\sigma')\lambda'} s\lambda'$$

où $\Phi_1\sigma' \subseteq \Phi'$*.*

Démonstration. D'après les lemmes précédents 4.15 et 4.24 et par induction sur la dérivation, il suffit de vérifier la propriété sur le système Σ.

Tout d'abord, rappelons que d'après la proposition 4.16, $t_0 \notin \mathrm{st}(\mathcal{N})$ implique $t_0 \notin \mathrm{st}(\mathrm{nps}(\mathcal{C}))$, donc le remplacement $\{t_0 \mapsto r\}$ n'opère pas dans $\mathcal{C}_\rhd$. Comme Σ_0 est pré-résolu, on en déduit $\mathrm{nps}(\mathcal{C}\{t_0 \mapsto r\}) = \mathrm{nps}(\mathcal{C}) = \emptyset$.

Or, du fait que $\mu = \mathrm{mgu}(t_0, f(t_1, \ldots, t_n))$, on a $\mathrm{supp}(\mu)\mu \subseteq \mathrm{st}(t_1\mu, \ldots, t_n\mu)$. Par conséquent le lemme 4.14 donne

$$\mathrm{nps}(\Sigma) \subseteq \mathrm{supp}(\mu)\mu \subseteq \mathrm{st}(t_1\mu, \ldots, t_n\mu) = \mathrm{st}(t_1\sigma', \ldots, t_n\sigma').$$

Enfin, soit $Y \in \mathrm{vnps}(\Sigma)$ quelconque et $(M \rhd t) \in \Phi_0$. Par le lemme 4.23 et la remarque précédente, il existe $y \in \mathrm{var}(\mu)$ tel que $(Y \rhd^? y) \in \mathcal{C}$.

Comme $\mathrm{var}(\mu) \subseteq \mathrm{var}(t_0, l)$ et $\mathrm{var}(l) \cap \mathrm{var}(\Sigma) = \emptyset$, on a $y \in \mathrm{var}(t_0) \subseteq \mathrm{var}(t)$. Σ étant stratifié (proposition 4.19), il existe $X \rhd^? s$ dans $\mathcal{C}$ tel que $X \preceq M$ et $y \in \mathrm{var}(s)$. Mais Σ est pré-résolu, donc $s = y$ et finalement $Y = X \preceq M$.

Concernant le dernier point, du fait que $\theta, \lambda' \models \Sigma'$, on a

$$X\lambda' \rhd_{\Phi'\lambda'} t\lambda'$$

Toute règle de Φ' intervenant dans cette relation s'écrit $(M \rhd t')\sigma'$ où $(M \rhd t') \in \Phi\{t_0 \mapsto r\}$ et où $(M\sigma')\lambda' = M\lambda'$ est un sous-terme de $X\lambda'$.

Or, d'une part, $\mathrm{maxpar}(M) \leq \mathrm{ar}(X)$. Dans l'hypothèse où $k_1 < \infty$, soit $M_0 \rhd t'_0$ dans Φ_0 tel que $\max(\mathrm{maxpar}(M_0), \mathrm{maxar}(M_0)) = k_1$. Comme $X \in \mathrm{vnps}(\Sigma')$, on a $X \preceq M_0\sigma'$, c'est-à-dire $\mathrm{ar}(X) < \mathrm{maxpar}(M_0\sigma')$ ou $X \in \mathrm{var}(M_0\sigma')$. Comme σ' est bien formée, on obtient $\mathrm{ar}(X) \leq \mathrm{maxpar}(M_0)$ ou $\mathrm{ar}(X) \leq \mathrm{maxar}(M_0)$; soit dans tous les cas, $\mathrm{maxpar}(M) \leq \mathrm{ar}(X) \leq k_1$.

D'autre part, si $(M \rhd t') \in \Phi_0$, alors $(M\sigma')\lambda' = M\lambda'$ est un sous-terme de $X\lambda'$ ce qui par le lemme 4.17 contredit le fait $X \preceq M\sigma'$.

Par conséquent, $(M \rhd t') \in \Phi_1$. Finalement, on a montré $X\lambda' \rhd_{(\Phi_1\sigma')\lambda'} t\lambda'$. □

Règles d'environnement générées

Nous précisons maintenant la forme des règles d'environnements engendrées par une dérivation standard.

Proposition 4.28. *Soit* Σ_0 *un système initial et* $\Sigma_0 \Longrightarrow^* \Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ *une dérivation aboutissant au système* Σ *et respectant l'ordre de priorité entre règles. Soit* $M \rhd t$ *une règle d'environnement de* Φ*. Alors, l'une des deux conditions suivantes est vraie :*

(i) $\mathrm{maxar}(M) < \mathrm{maxpar}(M)$ *(en particulier* $\mathrm{par}(M) \neq \emptyset$*) ;*

(ii) $M \rhd t$ *est la dernière règle créée par* **Context-1***, et* Σ *n'est pas pré-résolu.*

Définition 4.12. Un terme M ou une règle $M \rhd t$ vérifiant la condition (i) ci-dessus, $\mathrm{maxar}(M) < \mathrm{maxpar}(M)$, est dit(e) *totalement stratifié(e).*

En effet, on remarque que le pré-ordre de stratification $\preceq$ est total sur l'ensemble des termes totalement stratifiés : pour de tels termes M et N, la condition $M \preceq N$ équivaut en effet à $\mathrm{maxpar}(M) \leq \mathrm{maxpar}(N)$.

Démonstration de la proposition 4.28. Tout d'abord notons que la condition (i) est stable par substitution bien formée. Ceci entraîne que la propriété $\forall (M \rhd t) \in \Phi,\ \mathrm{maxar}(M) < \mathrm{maxpar}(M)$ est stable par toutes les règles de transformation excepté **Context-1**. Si $\Sigma = \Sigma_0$ ou si Σ dérive de Σ_0 sans utiliser de règle **Context-1**, la propriété est donc claire.

Soit $\Sigma_0 \Longrightarrow^* \Sigma_1$ une dérivation telle que $\Sigma_1 = \Phi_1; \Psi_1; \mathcal{C}_1; \mathcal{N}_1; \sigma_1$ est pré-résolu. Supposons que

$$\Sigma_1 \Longrightarrow_{\mathbf{Context-1}} \Sigma_2 = (\Phi_1 \cup \{M' \rhd r\};\ \Psi_1 \cup \{\forall Z_1, \ldots, Z_q.M \bowtie M'\};\ \mathcal{C}_1 \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\};\ \mathcal{N}_1 \cup \{l_1, \ldots, l_n\};\ \sigma_1)\mu$$

où $M' = M\{Z_j \mapsto \mathsf{c}_{\mathrm{sort}(Z_j)}\} = D[M_1, \ldots, M_n, Y_1, \ldots, Y_p, \mathsf{c}_{\mathrm{sort}(Z_1)}, \ldots, \mathsf{c}_{\mathrm{sort}(Z_q)}]$ avec les notations de la règle. Supposons enfin $\Sigma_1 \Longrightarrow^* \Sigma$ par une dérivation utilisant uniquement les règles de pré-résolution de telle sorte que $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ est le premier système pré-résolu rencontré.

Nous vérifions maintenant que si les règles d'environnement de Φ_1 sont totalement stratifiées, alors c'est le cas des règles de Φ. La proposition en découle par récurrence sur la longueur de la dérivation $\Sigma_0 \Longrightarrow^* \Sigma$, et par définition des dérivations respectant les priorités entre règles.

Tout d'abord, d'après le lemme 4.21, du fait des contraintes $Y_i\mu \rhd^? y_p\mu$ dans Σ_2, on a pour tout $1 \leq i \leq p$, $Y_i\sigma = Y_i\mu\sigma \rhd_{\Phi,\mathcal{C}} l_i\mu\sigma = l_i\sigma$. Il existe donc des contextes publics C_i, des règles $N_1 \rhd s_1, \ldots, N_m \rhd s_m$ dans Φ, et des contraintes $X_1 \rhd^? x_1, \ldots, X_r \rhd^? x_r$ dans $\mathcal{C}$ tels que les variables $X_1, \ldots, X_r, x_1, \ldots, x_r$ soient deux à deux disjointes et que pour tout i l'on ait

$$\begin{aligned} Y_i\sigma &= C_i[N_1, \ldots, N_m, X_1, \ldots, X_r] \\ l_i\sigma &= C_i[s_1, \ldots, s_m, x_1, \ldots, x_r] \end{aligned}$$

Quitte à restreindre et renuméroter les $N_i \rhd s_i$ et les $X_j \rhd^? x_j$, on peut supposer sans perte de généralité que $\bigcup_{i=1}^{p} \mathrm{par}(C_i) = \{w_1, \ldots, w_{m+r}\}$.

Posons

$$C = D[w_{m+r+1}, \ldots, w_{m+r+n}, C_1, \ldots, C_p, \mathsf{c}_{\mathrm{sort}(Z_1)}, \ldots, \mathsf{c}_{\mathrm{sort}(Z_q)}]$$

On a ainsi

$$M'\sigma = C[N_1, \ldots, N_m, X_1, \ldots, X_r, M_1\sigma, \ldots, M_n\sigma] \quad (4.3.1)$$

$$l\sigma = C[s_1, \ldots, s_m, x_1, \ldots, x_r, t_1\sigma, \ldots, t_n\sigma] \quad (4.3.2)$$

De plus, sachant que $\mathrm{par}(D) = \{w_1, \ldots, w_{n+p+q}\}$ par définition des décompositions, on a

$$\begin{aligned} \mathrm{par}(C) &= \bigcup_{i=1}^{p} \mathrm{par}(C_i) \cup \{w_{m+r+1}, \ldots, w_{m+r+n}\} \\ &= \{w_1, \ldots, w_{m+r+n}\} \end{aligned}$$

Nous montrons que la règle $M'\sigma \rhd r\sigma$ est totalement stratifiée, en vérifiant que pour tout $X_j \in \mathrm{var}(M'\sigma)$ $(1 \leq j \leq r)$,

$$\mathrm{ar}(X_j) < \mathrm{maxpar}(N_1, \ldots, N_m, M_1\sigma, \ldots, M_m\sigma) = \mathrm{maxpar}(M'\sigma).$$

Tout d'abord, notons que par l'hypothèse sur Σ_1, tous les termes $N_1, \ldots, N_m$, $M_1\sigma, \ldots, M_m\sigma$ sont totalement stratifiées (car apparaissant dans $\Phi \subseteq \Phi_1\sigma$).

Soit $X_j \in \mathrm{var}(M'\sigma)$. Si X_j apparaît dans un des $M_i\sigma$ sous-terme de $M'\sigma$, alors $\mathrm{ar}(X_j) < \mathrm{maxpar}(M_i\sigma) = \mathrm{maxpar}(M'\sigma)$

Dans le cas contraire, d'après l'équation 4.3.1, il existe $1 \leq i \leq p$ tel que

$$X_j \in \mathrm{var}(C_i[N_1, \ldots, N_m, X_1, \ldots, X_r]).$$

Si $X_j \in \mathrm{var}(N_1, \ldots, N_m)$, on a bien $\mathrm{ar}(X_j) < \mathrm{maxpar}(N_1, \ldots, N_m) = \mathrm{maxpar}(M'\sigma)$.

Sinon, x_j apparaît dans $l_i\sigma = C_i[s_1, \ldots, s_m, x_1, \ldots, x_r]$. Or, $\sigma = \sigma_1\mu\sigma$ et $l_i\sigma_1\mu = l_i\mu = t_i\mu = t_i\sigma_1\mu$. On a donc $l_i\sigma = t_i\sigma$ et $x_j \in \mathrm{var}(t_i\sigma)$. De plus, par définition des règles de transformation, $(M_i \rhd t_i) \in \Phi_1$ implique $(M_i \rhd t_i)\sigma \in \Phi$.

La propriété de stratification sur Σ (proposition 4.19) implique donc l'existence de $(X' \rhd^? t') \in \mathcal{C}$ tel que $x_j \in \mathrm{var}(t')$ et $X' \preceq M_i\sigma$. Mais Σ est pré-résolu, donc $x_j = t'$ et $X_j = X' \preceq M_i\sigma$. M_i et par conséquent $M_i\sigma$ étant totalement stratifié, ceci implique $\mathrm{ar}(X_j) < \mathrm{maxpar}(M_i\sigma) \leq \mathrm{maxpar}(M'\sigma)$. □

Équations générées

Dans le même ordre d'idée que le paragraphe précédent, nous précisons la forme des équations engendrées par une dérivation standard.

Proposition 4.29. *Soit Σ_0 un système initial et $\Sigma_0 \Longrightarrow^* \Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ une dérivation aboutissant au système Σ et respectant l'ordre de priorité entre règles. Soit $\forall\beta.M \bowtie N$ une équation dans Ψ. Alors, l'une des deux conditions suivantes est réalisée :*

(i) l'équation $\forall\beta.M \bowtie N$ est totalement stratifiée *au sens où pour toute variable $X \in \mathrm{var}(\forall\beta.M \bowtie N) = \mathrm{var}(M, N) - \beta$, on a $\mathrm{ar}(X) < \mathrm{maxpar}(M, N)$.*

(ii) $\forall\beta.M \bowtie N$ est l'équation venant d'être créée par **Context-{1,2}** *ou* **Relate**, *et Σ n'est pas pré-résolu.*

Démonstration. Tout d'abord notons que la condition (i) est stable par substitution bien formée, et donc a priori par toutes les règles des tables 4.1 et 4.2 à l'exception de **Context-{1,2}**, **Relate** et **Discard**.

Concernant la règle **Discard**, on note que si la règle source $M \rhd t$ est totalement stratifiée alors l'équation résultante $M \bowtie N$ l'est également, car $N \preceq M$.

Le cas de **Context-{1,2}** se déduit de la preuve de la proposition 4.28 : en effet, nous avons montré dans le cas de **Context-1** que la règle ajoutée $M' \rhd t$ est instanciée en une règle stratifiée après la phase de pré-résolution : avec les notations de la preuve, $\mathrm{maxar}(M'\sigma) < \mathrm{maxpar}(M'\sigma)$ (σ représente ici la substitution après pré-résolution). Par définition de M et M', on en déduit que

l'équation ajoutée $\forall\beta.M \bowtie M'$ où $\beta = \{Z_1, \ldots, Z_q\}$ vérifie

$$\begin{aligned} \mathrm{maxar}((\forall\beta.M \bowtie M')\sigma) &= \mathrm{maxar}(M'\sigma) \\ &< \mathrm{maxpar}(M'\sigma) \\ &= \mathrm{maxpar}((\forall\beta.M \bowtie M')\sigma) \end{aligned}$$

Le cas de la règle **Context-2** se traite de manière identique.

Il reste donc à considérer le cas des équations créés par **Relate**. Supposons $\Sigma_0 \Longrightarrow^* \Sigma_1 = \Phi_1;\ \Psi_1;\ \mathcal{C}_1;\ \mathcal{N}_1;\ \sigma_1$ où Σ_1 est pré-résolu et où les règles et les équations de Φ_1 sont totalement stratifiées (par la proposition 4.28 et par hypothèse de récurrence). Supposons de plus

$$\Sigma_1 \Longrightarrow_{\mathbf{Relate}} \Sigma_2 = \Phi_1;\ \Psi_1 \cup \{X \bowtie M\};\ \mathcal{C}_1 \cup \{X \rhd^? t\};\ \mathcal{N}_1 \cup \{t\};\ \sigma_1$$

et enfin $\Sigma_2 \Longrightarrow^* \Sigma$ par une dérivation utilisant les règles de pré-résolution uniquement. Nous vérifions que si Σ est pré-résolu, alors $X\sigma \bowtie M\sigma$ est totalement stratifiée.

En effet, par le lemme 4.21, on a $X\sigma \rhd_{\Phi,\mathcal{C}} t\sigma$. Sachant que Σ est pré-résolu, il existe donc un contexte public C, des règles $M_1 \rhd t_1$, …, $M_m \rhd t_m$ dans Φ, et des contraintes $X_1 \rhd^? x_1$, …, $X_n \rhd^? x_n$ dans $\mathcal{C}$ tels que

$$\begin{aligned} X\sigma &= C[M_1, \ldots, M_m, X_1, \ldots, X_n] \\ t\sigma &= C[t_1, \ldots, t_m, x_1, \ldots, x_n] \end{aligned}$$

Par la proposition 4.19, Σ est stratifié : pour tout i, $(M\sigma \rhd t\sigma) \in \Phi$ et $x_i \in \mathrm{var}(t\sigma)$ impliquent qu'il existe une contrainte $Z \rhd^? x_i$ dans $\mathcal{C}$ telle que $Z \preceq M\sigma$. Or Σ est pré-résolu, donc $X_i = Z \preceq M\sigma$.

Finalement, comme les règles de $\Phi = \Phi_1\sigma \supseteq \{M_1 \rhd t_1, \ldots, M_m \rhd t_m, M\sigma \rhd t\sigma\}$ sont totalement stratifiées, l'équation $X\sigma \bowtie M\sigma$ l'est donc également. □

Finalement, les propositions précédentes motivent la définition suivante.

Définition 4.13. Un système Σ est *totalement stratifié* ss'il est stratifié (au sens de la proposition 4.19), et que toutes les règles d'environnement et les équations de Σ sont totalement stratifiées.

Ainsi, les propositions précédentes impliquent en particulier que tout système Σ pré-résolu et accessible par une dérivation standard (ou simplement par une dérivation respectant l'ordre de priorité entre règles) est totalement stratifié.

4.4 Complétude des dérivations standard

Nous établissons maintenant la complétude des dérivations standard pour toute système de réécriture sous-terme-convergent.

Proposition 4.30 (Complétude)**.** *Supposons le système de réécriture $\mathcal{R}$ sous-terme-convergent. Soit Σ_0 un système de contraintes initial et θ une solution de Σ_0. Alors il existe un système de contraintes résolu Σ tel que $\theta \models \Sigma$ et $\Sigma_0 \Longrightarrow^* \Sigma$ par une dérivation standard.*

La preuve s'articule de la manière suivante. Tout d'abord, θ étant fixé, nous décrivons un ordre strict bien fondé $>_c$ sur les paires (λ, Σ) vérifiant $\theta, \lambda \models \Sigma$.

Ensuite, pour chaque règle **R**, nous présentons un lemme technique permettant de montrer, pour certains triplets $\theta, \lambda \models \Sigma$, l'existence de λ' et Σ' tels que

- $\Sigma \Longrightarrow_{\mathbf{R}} \Sigma'$,
- $\theta, \lambda' \models \Sigma'$,
- $(\theta, \lambda, \Sigma) >_c (\theta, \lambda', \Sigma')$ si **R** est une règle non prioritaire (à savoir **Narrowing**, **Constrain**, **Context-{1,2}** ou **Relate**),
- $(\theta, \lambda, \Sigma) \geq_c (\theta, \lambda', \Sigma')$ dans le cas contraire.

Cette distinction entre règles est justifiée par le fait que les séquences de règles prioritaires (**Project**, **Imitate**, **Coalesce**, **Fail** et **Discard**) terminent au niveau syntaxique (voir la sous-section 4.1.3).

Nous prouvons ensuite un résultat de progression montrant que pour tout triplet $\theta, \lambda \models \Sigma$ tel que Σ n'est pas résolu, il existe une règle de transformation — différente de **Fail** — pouvant s'appliquer au sens ci-dessus, ce d'une manière compatible avec les priorités des règles de dérivations standard.

Nous concluons enfin la preuve de complétude des dérivations standard à l'aide (notamment) de la proposition 4.9 concernant la condition d'arrêt $\geq$ (voir la section 4.1).

Remarque. Dans le cas des systèmes sous-terme-convergents, l'emploi de la mesure $>_c$ est en partie redondant avec la preuve de terminaison syntaxique (section 4.5). Toutefois, nous trouvons plus instructif de prouver la propriété de complétude indépendamment de la terminaison.

4.4.1 Mesure de terminaison sémantique

Soit $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ un système de contraintes et λ une substitution partielle close de domaine $\mathrm{var}(\Sigma)$.

Nous décrivons maintenant la mesure utilisée sur de tels couples (λ, Σ). Cette mesure comporte plusieurs composantes ayant pour but de contrôler intuitivement

- le nombre d'étapes de réductions par $\mathcal{R}$ dans les termes du premier ordre de Σ,
- le nombre de règles non totalement stratifiées,
- le nombre d'équations restant à ajouter par la règle **Relate** pour garantir (en particulier) la complétude de la règle **Coalesce**, et
- le nombre de contraintes d'égalité du système.

Réductions possibles

Soit $\mathrm{t}(\lambda, \Sigma)$ le multi-ensemble des termes du premier ordre apparaissant au sommet des règles et des contraintes de $\Sigma\lambda$, et n'étant pas $\mathcal{R}$-réduit :

$$\begin{aligned}
\mathrm{t}(\lambda, \Sigma) \;=\; & \{t\lambda \mid (M \rhd t) \in \Phi, \quad t\lambda \text{ non } \mathcal{R}\text{-réduit}\}^+ \\
\uplus\; & \{t\lambda \mid (X \rhd^? t) \in \mathcal{C}, \quad t\lambda \text{ non } \mathcal{R}\text{-réduit}\}^+ \\
\uplus\; & \{t_i\lambda \mid (t_1 =^?_E t_2) \in \mathcal{C}, \quad i \in \{1,2\}, \quad t_i\lambda \text{ non } \mathcal{R}\text{-réduit}\}^+ \\
\uplus\; & \{t_i\lambda \mid (t_1 \neq^?_E t_2) \in \mathcal{C}, \quad i \in \{1,2\}, \quad t_i\lambda \text{ non } \mathcal{R}\text{-réduit}\}^+
\end{aligned}$$

Soit $\to_{\mathcal{R}}^* \succeq_{\mathrm{st}}$ la relation défini par $t \to_{\mathcal{R}}^* \succeq_{\mathrm{st}} t'$ ss'il existe t'' tel que $t' \to_{\mathcal{R}}^* t''$ et $t' \in \mathrm{st}(t'')$. Du fait de l'inclusion $(\succeq_{\mathrm{st}} \to_{\mathcal{R}}) \subseteq (\to_{\mathcal{R}} \succeq_{\mathrm{st}})$, cette relation est un pré-ordre (*i.e.* relation réflexive et transitive).

On munit l'ensemble des $\mathrm{t}(\lambda, \Sigma)$ du pré-ordre $(\to_{\mathcal{R}}^* \succeq_{\mathrm{st}})_{\mathrm{mul}}$ défini comme l'extension de $\to_{\mathcal{R}}^* \succeq_{\mathrm{st}}$ aux multi-ensembles de termes de la manière habituelle [CJ03].

Comme $>_{\mathrm{st}}$ et $\to_{\mathcal{R}}^+$ terminent et l'on a l'inclusion $(>_{\mathrm{st}} \to_{\mathcal{R}}) \subseteq (\to_{\mathcal{R}} >_{\mathrm{st}})$, le pré-ordre strict associé à $\to_{\mathcal{R}}^* \succeq_{\mathrm{st}}$, c'est-à-dire $(>_{\mathrm{st}}) \cup (\to_{\mathcal{R}}^+ \succeq_{\mathrm{st}})$, est bien terminant (voir [Der87] pour une généralisation de cette propriété). Par conséquent, l'ordre strict associé à $(\to_{\mathcal{R}}^* \succeq_{\mathrm{st}})_{\mathrm{mul}}$ termine également.

Règles non totalement stratifiées

Notons Φ^{S} l'ensemble des règles d'environnement $M \rhd t$ de Φ telles que M est totalement stratifié (*i.e.* $\mathrm{maxar}(M) < \mathrm{maxpar}(M)$).

Soit $\mathrm{s}(\Sigma)$ le cardinal de $\Phi - \Phi^{\mathrm{S}}$. On remarque que d'après la proposition 4.28, $\mathrm{s}(\Sigma) \in \{0, 1\}$ pour tout Σ accessible par une dérivation standard.

Équations manquantes

Pour tout terme clos $\mathcal{R}$-réduit t, on considère l'ensemble $\mathrm{R}(\lambda, \Sigma, t)$ des quadruplets (M_1, t_1, M_2, t_2) vérifiant les conditions suivantes :

- $(M_1 \rhd t_1) \in \Phi^{\mathrm{S}}$ et $M_2 \rhd_{\Phi^{\mathrm{S}}} t_2$;
- $t_1\lambda = t_2\lambda = t$;
- Il n'existe pas de sous-ensemble $\Psi_0 \subseteq \Psi$ tel que

$$M_1\lambda \bowtie_{\Psi_0\lambda} M_2\lambda \quad \text{et} \quad \mathrm{maxpar}(\Psi_0\lambda) \leq \mathrm{maxpar}(M_1\lambda, M_2\lambda).$$

Notons que les ensemble $\mathrm{R}(\lambda, \Sigma, t)$ sont finis, car d'une part Φ^{S} est fini, et d'autre part il existe un nombre fini de t_2 tels que $t_2\lambda = t$, puis un nombre fini de M_2 tels que $M_2 \rhd_{\Phi^{\mathrm{S}}} t_2$.

Soit $\mathrm{r}(\lambda, \Sigma, t)$ le cardinal de $\mathrm{R}(\lambda, \Sigma, t)$, et $\mathrm{r}(\lambda, \Sigma)$ la fonction associant à chaque terme clos $\mathcal{R}$-réduit t, le nombre $\mathrm{r}(\lambda, \Sigma, t)$.

Du fait que Φ^{S} est fini, à λ et Σ fixés, l'ensemble des termes t tels que $\mathrm{r}(\lambda, \Sigma, t) \neq 0$, c'est-à-dire le *support* de $\mathrm{r}(\lambda, \Sigma)$, est fini.

Considérons l'ensemble des fonctions, notées r, allant des termes clos dans les entiers positifs et à support fini. On munit cet ensemble de l'ordre point à point : $r \dot{\geq} r'$ ssi pour tout t, $r(t) \geq r'(t)$. L'ordre strict associé à $\dot{\geq}$ est noté $\dot{>}$. Il s'agit d'un ordre bien fondé.

Contraintes d'égalité

Finalement, soit $\mathrm{c}(\Sigma) = \mathrm{card}(\mathcal{C}_=)$ le nombre de contraintes d'égalité $t_1 =_E^? t_2$ dans $\mathcal{C}$.

Mesure obtenue

On définit la *mesure de* (λ, Σ) par

$$\mathrm{m}_c(\lambda, \Sigma) \triangleq (\mathrm{t}(\lambda, \Sigma),\ \mathrm{s}(\Sigma),\ \mathrm{r}(\lambda, \Sigma),\ \mathrm{c}(\Sigma)).$$

On note $(\lambda, \Sigma) \geq_c (\lambda', \Sigma')$ lorsque $\mathrm{m}_c(\lambda, \Sigma)$ est plus grand (au sens large) que $\mathrm{m}_c(\lambda', \Sigma')$ pour l'ordre lexicographique induit de gauche à droite par les pré-ordres $(\rightarrow_{\mathcal{R}}^{*} \succeq_{\mathrm{st}})_{\mathrm{mul}}$, $\geq$, $\dot{\geq}$, et $\geq$.

L'ordre strict associé à $\geq_c$ est noté $>_c$.

Autres notations et lemmes techniques

Pour exprimer certains invariants, nous utiliserons également la notation suivante : étant donnés des ensembles Φ_0 et Ψ_0, un terme t $\mathcal{R}$-réduit clos et une substitution λ de domaine contenant $\mathrm{var}(\Phi_0, \Psi_0)$, on note $\mathrm{R}^*(\lambda, \Phi_0, \Psi_0, t)$ l'ensemble des quadruplets (M_1, t_1, M_2, t_2) vérifiant

- $(M_1 \rhd t_1) \in \Phi_0$ et $M_2 \rhd_{\Phi_0} t_2$;
- $t_1\lambda = t_2\lambda = t$;
- Il n'existe pas de sous-ensemble $\Psi_1 \subseteq \Psi_0$ tel que

$$M_1\lambda \bowtie_{\Psi_1\lambda} M_2\lambda \quad \text{et} \quad \mathrm{maxpar}(\Psi_1\lambda) \leq \mathrm{maxpar}(M_1\lambda, M_2\lambda).$$

De cette façon, étant donnés $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ un système de contraintes et λ une substitution partielle close de domaine $\mathrm{var}(\Sigma)$, on a

$$\mathrm{R}(\lambda, \Sigma, t) = \mathrm{R}^*(\lambda, \Phi^{\mathrm{S}}, \Psi, t).$$

Concernant les notations Φ^{S}, $\mathrm{R}(\lambda, \Sigma, t)$ et $\mathrm{R}^*(\lambda, \Phi_0, \Psi_0, t)$, nous utilisons plus loin les lemmes techniques suivants.

Lemme 4.31. *Soit Φ un ensemble de règles d'environnement, σ une substitution (bien formée). Alors*

$$(\Phi\sigma)^{\mathrm{S}} \supseteq (\Phi^{\mathrm{S}})\sigma$$

De même, si Ψ est un ensemble d'équations totalement stratifiées, alors c'est le cas de $\Psi\sigma$.

Démonstration. Soit $M \rhd t$ une règle de Φ totalement stratifiée : $\mathrm{maxar}(M) < \mathrm{maxpar}(M)$. Comme σ est bien formée, on a

$$\mathrm{maxar}(M\sigma) \leq \mathrm{maxar}(M) < \mathrm{maxpar}(M) \leq \mathrm{maxpar}(M\sigma)$$

d'où $(M \rhd t)\sigma \in (\Phi\sigma)^{\mathrm{S}}$.

Le cas de $\Psi\sigma$ se prouve de manière identique. □

Lemme 4.32. *Soit Φ un ensemble de règles d'environnement, σ une substitution (bien formée), M' et t' deux termes tels que $M' \rhd_{\Phi\sigma} t'$. Alors il existe M et t tel que*

$$M' = M\sigma$$
$$t' = t\sigma$$
$$M \rhd_{\Phi} t$$

Démonstration. Par définition de la notation $\rhd$ et de $\Phi\sigma$, $M' \rhd_{\Phi\sigma} t'$ signifie qu'il existe un contexte public C et des règles $M_1 \rhd t_1, \ldots, M_n \rhd t_n$ dans Φ tels que

$$M' = C[M_1\sigma, \ldots, M_n\sigma] = C[M_1, \ldots, M_n]\sigma$$
$$t' = C[t_1\sigma, \ldots, t_n\sigma] = C[t_1, \ldots, t_n]\sigma$$

(C étant clos par définition). On pose $M = C[M_1, \ldots, M_n]$ et $t = C[t_1, \ldots, t_n]$. □

Lemme 4.33. *Soit Φ_1 un ensemble de règles totalement stratifiées et Ψ_1 un ensemble d'équations. Soit λ et λ' des substitutions partielles closes, σ' une substitution, et Ψ_0 un ensemble d'équations totalement stratifiées tels que*

(i) $\mathrm{dom}(\lambda') \supseteq \mathrm{dom}(\lambda) \supseteq \mathrm{var}(\Phi_1, \Psi_1, \Psi_0)$,

(ii) $\lambda' = \sigma'|_{\mathrm{dom}(\lambda')}\lambda'$,

(iii) pour tout $x \in \mathrm{var}(\Phi_1)$, $x\lambda' = x\lambda$,

(iv) pour tout $X \in \mathrm{var}(\Phi_1, \Psi_1, \Psi_0)$,
- $\mathrm{maxpar}(X\lambda') \leq \mathrm{maxpar}(X\lambda)$,
- $X\lambda' \bowtie_{\Psi_0\lambda'} X\lambda$,
- *si* $X\lambda' \neq X\lambda$, *alors* $\mathrm{maxpar}(X\lambda) \geq \mathrm{maxpar}(\Psi_0\lambda)$. *(En particulier, comme Ψ_0 est totalement stratifié, on a $\Psi_0\lambda' = \Psi_0\lambda$.)*

Alors pour tout t_0 $\mathcal{R}$-réduit et clos, on a

$$\mathrm{R}^*(\lambda', \Phi_1\,\sigma', (\Psi_0 \cup \Psi_1)\,\sigma', t_0) \subseteq \mathrm{R}^*(\lambda, \Phi_1, \Psi_1, t_0)\,\sigma'$$

Démonstration. Soit $(M_1', t_1', M_2', t_2') \in \mathrm{R}^*(\lambda', \Phi_1\sigma', (\Psi_0 \cup \Psi_1)\sigma', t_0)$. Comme $(M_1' \rhd t_1') \in \Phi_1\sigma'$ et $M_2' \rhd_{\Phi_1\sigma'} t_2'$, par le lemme 4.32 il existe (M_1, t_1, M_2, t_2) tel que $(M_1 \rhd t_1) \in \Phi_1$, $M_2 \rhd_{\Phi_1} t_2$ (en particulier $\mathrm{var}(M_1, t_1, M_2, t_2) \subseteq \mathrm{var}(\Phi_1)$) et $(M_1', t_1', M_2', t_2') = (M_1, t_1, M_2, t_2)\sigma'$. Notamment $t_1\lambda = t_1\lambda' = t_2\lambda' = t_2\lambda = t_0$ par (ii) et (iii), et pour $i \in \{1, 2\}$, $M_i'\lambda_i' = M_i\lambda_i'$ par (ii).

Supposons par contradiction que $(M_1, t_1, M_2, t_2) \notin \mathrm{R}^*(\lambda, \Phi_1, \Psi_1, t_0)$, c'est-à-dire qu'il existe $\Psi_2 \subseteq \Psi_1$ tel que $M_1\lambda \bowtie_{\Psi_2\lambda} M_2\lambda$ et $\mathrm{maxpar}(\Psi_2\lambda) \leq \mathrm{maxpar}(M_1\lambda, M_2\lambda)$.

(1) Si $\mathrm{maxpar}(\Psi_0\lambda) > \mathrm{maxpar}(M_1\lambda, M_2\lambda)$, alors

$$\mathrm{maxpar}(\Psi_0\lambda) > \mathrm{maxpar}(M_1\lambda, M_2\lambda) \geq \mathrm{maxpar}(\Psi_2\lambda)$$

implique $\Psi_2\lambda = \Psi_2\lambda'$, $M_1\lambda' = M_1\lambda$ et $M_2\lambda' = M_2\lambda$. On en déduit

$$M_1'\lambda' = M_1\lambda \bowtie_{\Psi_2\lambda'} M_2\lambda = M_2'\lambda' \text{ et } \mathrm{maxpar}(\Psi_2\lambda') \leq \mathrm{maxpar}(M_1\lambda', M_2\lambda'),$$

ce qui contredit le fait que $(M_1', t_1', M_2', t_2') \in \mathrm{R}^*(\lambda', \Phi_1\sigma', (\Psi_0 \cup \Psi_1)\sigma', t_0)$.

(2) Sinon, on a $\mathrm{maxpar}(\Psi_0\lambda) \leq \mathrm{maxpar}(M_1\lambda, M_2\lambda)$. Posons $\Psi_2' = \Psi_0 \cup \Psi_2$. On a $\Psi_2' \subseteq \Psi_0 \cup \Psi_1$. De plus, les règles de Ψ_2' vérifient

$$\begin{array}{rcl} \mathrm{maxpar}(\Psi_2'\lambda') & \leq & \mathrm{maxpar}(\Psi_2'\lambda) \\ & & \quad \text{par construction de } \lambda' \\ & \leq & \mathrm{maxpar}(M_1\lambda, M_2\lambda) \\ & & \quad \text{par définition de } \Psi_2 \text{ et par l'hypothèse du (2)} \\ & = & \mathrm{maxpar}(M_1\lambda', M_2\lambda') \\ & & \quad \text{car } M_1 \text{ et } M_2 \text{ sont totalement stratifiés} \end{array}$$

Or, du fait que pour tout X, $X\lambda' \bowtie_{\Psi_0\lambda'} X\lambda$, on a d'une part

$$(\bowtie_{\Psi_2\lambda}) \subseteq (\bowtie_{\Psi_2\lambda' \cup \Psi_0\lambda'}) = (\bowtie_{\Psi_2'\lambda'})$$

et d'autre part

$$M'_1\lambda' = M_1\lambda' \bowtie_{\Psi_0\lambda} M_1\lambda \bowtie_{\Psi_2\lambda} M_2\lambda \bowtie_{\Psi_0\lambda} M_2\lambda' = M'_2\lambda'$$

D'où $M'_1\lambda' \bowtie_{\Psi'_2\lambda'} M'_2\lambda'$ ce qui contredit finalement le fait que

$$(M'_1, t'_1, M'_2, t'_2) \in \mathrm{R}^*(\lambda', \Phi_1\sigma', (\Psi_0 \cup \Psi_1)\sigma', t_0). \qquad \square$$

4.4.2 Réduction des solutions d'un système

Nous décrivons maintenant les lemmes de réduction correspondants à chaque règle de transformation.

D'une manière générale, ces lemmes s'appliquent aux systèmes Σ *accessibles*, c'est-à-dire tels qu'il existe un système de contrainte initial Σ_0 tel que $\Sigma_0 \Longrightarrow^* \Sigma$.

Lemme 4.34 (Project). *Soit Σ un système accessible. Supposons les conditions suivantes vérifiées :*

- $\theta, \lambda \models \Sigma$;
- $\Sigma = \Phi;\ \Psi;\ \mathcal{C} \uplus \{X \rhd^? f(t_1, \ldots, t_n)\};\ \mathcal{N};\ \sigma$;
- *il existe une règle totalement stratifiée $(M \rhd t) \in \Phi^{\mathrm{S}}$ telle que $X\lambda = M\lambda$ et $f(t_1, \ldots, t_n)\lambda = t\lambda$.*

Alors

- $X \notin \mathrm{var}(M)$;
- $\mathrm{maxpar}(M) \leq \mathrm{ar}(X)$;
- $\mathrm{maxar}(M) \leq \mathrm{ar}(X)$;
- *il existe* $\mu = \mathrm{mgu}(t, f(t_1, \ldots, t_n))$.

Soit

$$\Sigma' = (\ \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma\ )\,\{X \mapsto M\}\mu.$$

On a $\Sigma \Longrightarrow_{\mathbf{Project}} \Sigma'$, et $\lambda' = \lambda$ vérifie

- $\theta, \lambda' \models \Sigma'$,
- $(\lambda, \Sigma) \geq_c (\lambda', \Sigma')$, *et*
- *Pour tous $\Phi_1 \subseteq \Phi$, $\Psi_1 \subseteq \Psi$ et t_1 $\mathcal{R}$-réduit clos,*

$$\mathrm{R}^*(\lambda', \Phi_1\,\sigma', \Psi_1\,\sigma', t_1) \subseteq \mathrm{R}^*(\lambda, \Phi_1, \Psi_1, t_1)\,\sigma'$$

Démonstration. La condition $X \notin \mathrm{var}(M)$ découle de l'hypothèse $X\lambda = M\lambda$ et de l'invariant $M \notin \mathcal{X}$ (proposition 4.19). La condition $\mathrm{maxpar}(M) \leq \mathrm{ar}(X)$ provient de la bonne formation de λ. On en déduit $\mathrm{maxar}(M) \leq \mathrm{ar}(X)$ car M est totalement stratifié.

De plus, $t\lambda = f(t_1, \ldots, t_n)\lambda$ implique que t et $f(t_1, \ldots, t_n)$ sont unifiables et $\mu\lambda = \lambda$. Ainsi, on a bien d'une part $\Sigma \Longrightarrow_{\mathbf{Project}} \Sigma'$ et d'autre part $\theta, \lambda' \models \Sigma'$ du fait que $\theta, \lambda \models \Sigma$ et $\{X \mapsto M\}\mu\lambda = \lambda$.

Par ailleurs, comme $\Sigma\lambda$ et $\Sigma'\lambda'$ ne diffèrent que de la contrainte $(X \rhd^? f(t_1, \ldots, t_n))\lambda$ supprimée dans $\Sigma'\lambda$, on a $\mathrm{t}(\lambda', \Sigma') \leq \mathrm{t}(\lambda, \Sigma)$. (En fait, il y a égalité car $f(t_1, \ldots, t_n) \in \mathrm{st}(\mathcal{N})$ par la proposition 4.16.) Clairement $\mathrm{c}(\Sigma') = \mathrm{c}(\Sigma)$, et par le lemme 4.31, $\mathrm{s}(\Sigma') \leq \mathrm{s}(\Sigma)$.

Enfin, compte tenu de $\lambda' = \lambda$, le troisième point : pour tous $\Phi_1 \subseteq \Phi$, $\Psi_1 \subseteq \Psi$ et t_1 $\mathcal{R}$-réduit clos,

$$\mathrm{R}^*(\lambda', \Phi_1\,\sigma', \Psi_1\,\sigma', t_1) \subseteq \mathrm{R}^*(\lambda, \Phi_1, \Psi_1, t_1)\,\sigma'$$

découle du lemme 4.33 où l'on prend $\Psi_0 = \emptyset$. Ceci implique en particulier, dans le cas où $\mathrm{s}(\Sigma') = \mathrm{s}(\Sigma)$ (*i.e.* $(\Phi')^{\mathrm{S}} = \Phi^{\mathrm{S}}\sigma'$), et en prenant $\Phi_1 = \Phi^{\mathrm{S}}$ et $\Psi_1 = \Psi$: pour tout t, $R(\lambda', \Sigma', t) \subseteq R(\lambda, \Sigma, t)\,\sigma'$. Ainsi on obtient $\mathrm{r}(\lambda', \Sigma') \dot{\geq} \mathrm{r}(\lambda, \Sigma)$ et finalement $(\lambda, \Sigma) \geq_c (\lambda', \Sigma')$. □

Lemme 4.35 (**Imitate**). *Soit Σ un système accessible. Supposons les conditions suivantes vérifiées :*

- $\theta, \lambda \models \Sigma$;
- $\Sigma = \Phi;\ \Psi;\ \mathcal{C} \uplus \{X \rhd^? f(t_1, \ldots, t_n)\};\ \mathcal{N};\ \sigma$;
- *il existe des règles $M_1 \rhd s_1$, …, $M_m \rhd s_m$ dans Φ et un contexte public $C = f(C_1, \ldots, C_n)$ tels que*

$$X\lambda = C[M_1, \ldots, M_m]\lambda \quad \textit{et} \quad f(t_1, \ldots, t_n)\lambda = C[s_1, \ldots, s_m]\lambda.$$

Alors $f \in \mathcal{F}_{pub}$. Soit $X_1, \ldots, X_n$ des variables fraîches d'arité $\mathrm{ar}(X)$*, et*

$$\Sigma' = (\,\Phi;\ \Psi;\ \mathcal{C} \cup \{X_1 \rhd^? t_1, \ldots, X_n \rhd^? t_n\};\ \mathcal{N};\ \sigma\,)\,\{X \mapsto f(X_1, \ldots, X_n)\}.$$

On a $\Sigma \Longrightarrow_{\mathbf{Imitate}} \Sigma'$ et la substitution

$$\lambda' = \lambda \uplus \{X_i \mapsto C_i[M_1, \ldots, M_m]\lambda\}_{1 \leq i \leq n}$$

vérifie

- $\theta, \lambda' \models \Sigma'$,
- $(\lambda, \Sigma) \geq_c (\lambda', \Sigma')$, *et*
- *pour tous $\Phi_0 \subseteq \Phi$, $\Psi_0 \subseteq \Psi$ et t_0 $\mathcal{R}$-réduit clos,*

$$\mathrm{R}^*(\lambda', \Phi_0\,\sigma', \Psi_0\,\sigma', t_0) \subseteq \mathrm{R}^*(\lambda, \Phi_0, \Psi_0, t_0)\,\sigma'$$

Démonstration. On a bien $\Sigma \Longrightarrow_{\mathbf{Imitate}} \Sigma'$ et $\theta, \lambda' \models \Sigma'$ par construction. De plus, $\mathrm{t}(\lambda, \Sigma) = \mathrm{t}(\lambda', \Sigma')$ car $f(t_1, \ldots, t_n)\lambda$ est $\mathcal{R}$-réduit (proposition 4.16). L'égalité $c(\Sigma) = c(\Sigma')$ est claire, de même que $\mathrm{s}(\Sigma') \leq \mathrm{s}(\Sigma)$ par le lemme 4.31. Enfin, $\mathrm{r}(\lambda, \Sigma) \dot{\geq} \mathrm{r}(\lambda', \Sigma')$ et le troisième point se montrent de manière similaire au cas de **Project** à l'aide du lemme 4.33. □

Lemme 4.36 (**Coalesce**). *Soit Σ un système accessible. Supposons les conditions suivantes vérifiées :*

- $\theta, \lambda \models \Sigma$;
- $\Sigma = \Phi;\ \Psi;\ \mathcal{C} \uplus \{X_1 \rhd^? t, X_2 \rhd^? t\};\ \mathcal{N};\ \sigma$;
- $\mathrm{ar}(X_1) \leq \mathrm{ar}(X_2)$.

De plus, supposons fixés des sous-ensembles de règles $\Phi_0 \subseteq \Phi$ et d'équations $\Psi_0 \subseteq \Psi$ tels que

- $X_1\lambda \rhd_{\Phi_0\lambda} t\lambda$ *et* $X_2\lambda \rhd_{\Phi_0\lambda} t\lambda$;
- *les équations de Ψ_0 sont totalement stratifiées ;*
- *pour tout $t_0 \in \mathrm{st}(t\lambda)$, $\mathrm{R}^*(\lambda, \Phi_0, \Psi_0, t_0) = \emptyset$.*

Posons

$$\Sigma' = (\,\Phi;\ \Psi;\ \mathcal{C} \cup \{X_1 \rhd^? t\};\ \mathcal{N};\ \sigma\,)\,\{X_2 \mapsto X_1\}.$$

Alors on a $\Sigma \Longrightarrow_{\mathbf{Coalesce}} \Sigma'$ et il existe λ' clos de domaine $\mathrm{var}(\Sigma') = \mathrm{var}(\Sigma)$ *tel que*

- $\theta, \lambda' \models \Sigma'$;

– $(\lambda, \Sigma) \geq_c (\lambda', \Sigma')$;
– *pour tout* t_1 $\mathcal{R}$*-réduit clos, pour tout* $\Phi_1 \subseteq \Phi^{\mathrm{S}}$ *et pour tout* $\Psi_1 \subseteq \Psi$*, on a*

$$\mathrm{R}^*(\lambda', \Phi_1\,\sigma', (\Psi_0 \cup \Psi_1)\,\sigma', t_1) \subseteq \mathrm{R}^*(\lambda, \Phi_1, \Psi_1, t_1)\,\sigma'$$

Démonstration. Notons tout d'abord que $t\lambda$ est bien $\mathcal{R}$-réduit du fait que $\theta, \lambda \models \Sigma$ et $t \in \mathrm{nps}(\Sigma) \subseteq \mathrm{st}(\mathcal{N})$ par la proposition 4.16.

Soit $\mathcal{C}_\rhd = \{Y_1 \rhd s_1, \ldots, Y_n \rhd^? s_n\}$ l'ensemble des contraintes de déductibilité dans $\mathcal{C}$. Comme $\theta, \lambda \models \Sigma$, il existe des contextes publics D_1, D_2, C_1, …, C_n, des règles $M_1 \rhd t_1$, …, $M_m \rhd t_m$ dans Φ tels que λ est solution du système d'équations $\mathcal{U}$:

$$\left\{\begin{array}{llll} Y_1 & =^? & C_1[M_1,\ldots,M_m] & s_1 =^? C_1[t_1,\ldots,t_m] \\ & \ldots & & \ldots \\ Y_n & =^? & C_n[M_1,\ldots,M_m] & s_n =^? C_n[t_1,\ldots,t_m] \\ X_1 & =^? & D_1[M_1,\ldots,M_m] & t =^? D_1[t_1,\ldots,t_m] \\ X_2 & =^? & D_2[M_1,\ldots,M_m] & t =^? D_2[t_1,\ldots,t_m] \end{array}\right.$$

Afin de construire λ' tel que $\theta, \lambda' \models \Sigma'$ (entre autres), nous distinguons plusieurs cas selon les valeurs de $X_1\lambda$ et $X_2\lambda$.
1. Si $X_1\lambda = X_2\lambda$, alors $\lambda' = \lambda$ satisfait les conditions voulues du fait de $\Sigma\lambda = \Sigma'\lambda'$ et du lemme 4.33 (le raisonnement est similaire au cas de **Project**).

On peut donc supposer $X_1\lambda \neq X_2\lambda$.
2. Si $\mathrm{maxpar}(X_1\lambda) \leq \mathrm{maxpar}(X_2\lambda)$, alors comme $X_2\lambda = D_2[M_1,\ldots,M_m]\lambda$ n'est pas un sous-terme de $X_1\lambda = D_1[M_1,\ldots,M_m]\lambda$, que λ est solution du système $\mathcal{U}_2$:

$$\left\{\begin{array}{lll} Y_1 & =^? & C_1[M_1,\ldots,M_m]\{X_2 \mapsto D_2[M_1,\ldots,M_m]\} \\ & \ldots & \\ Y_n & =^? & C_n[M_1,\ldots,M_m]\{X_2 \mapsto D_2[M_1,\ldots,M_m]\} \end{array}\right.$$

et que les $M_1, \ldots, M_m$ ne sont pas des variables (proposition 4.19), le lemme 4.18 implique l'existence d'un λ^1 solution du système $\mathcal{U}_2'$:

$$\left\{\begin{array}{lll} Y_1 & =^? & C_1[M_1,\ldots,M_m]\{X_2 \mapsto D_1[M_1,\ldots,M_m]\} \\ & \ldots & \\ Y_n & =^? & C_n[M_1,\ldots,M_m]\{X_2 \mapsto D_1[M_1,\ldots,M_m]\} \end{array}\right.$$

et vérifiant les conditions suivantes :

(a) $\mathrm{dom}(\lambda^1) = \mathrm{dom}(\lambda)$;
(b) $D_1[M_1,\ldots,M_m]\lambda^1 = D_1[M_1,\ldots,M_m]\lambda = X_1\lambda$;
(c) $D_2[M_1,\ldots,M_m]\lambda^1 = D_2[M_1,\ldots,M_m]\lambda = X_2\lambda$;
(d) pour tout $V \in \mathrm{dom}(\lambda) - \{Y_1,\ldots,Y_m\}$; $V\lambda^1 = V\lambda$, et
(e) pour tout $1 \leq i \leq n$,
  – $Y_i\lambda^1 \bowtie_{\{X_1\lambda \bowtie X_2\lambda\}} Y_i\lambda$,
  – $\mathrm{maxpar}(Y_i\lambda^1) \leq \mathrm{maxpar}(Y_i\lambda)$, et
  – si $Y_i\lambda^1 \neq Y_i\lambda$ alors $X_2\lambda$ est un sous-terme de $Y_i\lambda$.

Posons $\lambda' = \sigma\{X_2 \mapsto X_1\}\lambda^1$. Des équations précédentes, on déduit notamment que pour tout $X \in \mathrm{dom}(\Sigma)$,

$$X\lambda' \bowtie_{\{X_1\lambda \bowtie X_2\lambda\}} X\lambda,$$

de plus $X_1\lambda' = X_2\lambda' = X_1\lambda$ et pour tout $X \in \mathrm{var}(\Phi, \Psi, \mathcal{C}), X\lambda \neq X\lambda' \Rightarrow X_2\lambda \in \mathrm{st}(X\lambda)$.

Par construction de $\mathcal{U}'_2$, λ' satisfait les contraintes de déductibilité de Σ'. Comme λ' coïncide avec λ au premier ordre, il satisfait également les contraintes d'égalité et de différence modulo E. De plus, pour tout $V \in \mathrm{var}(\Sigma') = \mathrm{var}(\Sigma)$, $V\sigma'\lambda' = V(\sigma\{X_2 \mapsto X_1\})^2\lambda^1 = V\sigma\{X_2 \mapsto X_1\}\lambda^1 = V\lambda'$.

Pour avoir $\theta, \lambda' \models \Sigma'$, il reste à vérifier que pour tout $X \in \mathcal{Y}$, $X\theta \bowtie_{\Psi\lambda'} X\lambda'$ sachant que par hypothèse $X\theta \bowtie_{\Psi\lambda} X\lambda$.

De $t\lambda = D_1[t_1\lambda, \ldots, t_m\lambda] = D_2[t_1\lambda, \ldots, t_m\lambda]$, on déduit (par récurrence sur D_1 et D_2) qu'il existe un plus grand sur-contexte commun D à D_1 et D_2 :

$$\begin{aligned} D_1 &= D[w_{i^1_1}, \ldots, w_{i^1_p}, C^1_1, \ldots, C^1_q] \\ D_2 &= D[C^2_1, \ldots, C^2_p, w_{i^2_1}, \ldots, w_{i^2_q}] \end{aligned}$$

avec pour tous $1 \leq k \leq p$ et $1 \leq \ell \leq q$: $C^2_k, C^1_\ell \notin \mathcal{W}$ et $1 \leq i^2_k, i^1_\ell \leq m$. En particulier, pour tout k et ℓ, les termes

$$\begin{aligned} t_{i^1_k}\lambda &= C^2_k[t_1, \ldots, t_m]\lambda \\ t_{i^2_\ell}\lambda &= C^1_\ell[t_1, \ldots, t_m]\lambda \end{aligned}$$

sont des sous-termes de $t\lambda$. Or, on a $(M_{i^j_k} \rhd t_{i^j_k}) \in \Phi$ et

$$C^i_k[M_1, \ldots, M_m] \rhd_\Phi C^i_k[t_1, \ldots, t_m].$$

De plus, sachant que $X_1\lambda \rhd_{\Phi_0\lambda} t$ et $X_2\lambda \rhd_{\Phi_0\lambda} t$ par hypothèse, on peut choisir les indices $i \in \{i^2_k, i^1_\ell\}$ ci-dessus tel que tous les $M_i \rhd t_i$ soient dans Φ_0. Comme $\forall t_0 \in \mathrm{st}(t\lambda)$, $\mathrm{R}^*(\lambda, \Phi_0, \Psi_0, t_0) = \emptyset$, il existe donc des sous-ensembles d'équations $\Psi^1_k \subseteq \Psi_0$ et $\Psi^2_\ell \subseteq \Psi_0$ tels que

$$\begin{aligned} C^2_k[M_1, \ldots, M_m]\lambda &\bowtie_{\Psi^1_k\lambda} M_{i^1_k}\lambda \\ C^1_\ell[M_1, \ldots, M_m]\lambda &\bowtie_{\Psi^2_\ell\lambda} M_{i^2_\ell}\lambda \\ \mathrm{maxpar}(\Psi^1_k\lambda) &\leq \mathrm{maxpar}(C^2_k[M_1, \ldots, M_m]\lambda, M_{i^1_k}\lambda) \\ \mathrm{maxpar}(\Psi^2_\ell\lambda) &\leq \mathrm{maxpar}(C^1_\ell[M_1, \ldots, M_m]\lambda, M_{i^2_\ell}\lambda) \end{aligned}$$

En particulier, en posant $\Psi'_0 = \bigcup_{i,k} \Psi^i_k \subseteq \Psi_0$, on a d'une part

$$X_1\lambda = D_1[M_1, \ldots, M_m]\lambda \bowtie_{\Psi'_0\lambda} D_2[M_1, \ldots, M_m]\lambda = X_2\lambda$$

et d'autre part $\mathrm{maxpar}(\Psi'_0\lambda) \leq \mathrm{maxpar}(X_1\lambda, X_2\lambda) = \mathrm{maxpar}(X_2\lambda)$. Nous vérifions que $\Psi'_0\lambda = \Psi'_0\lambda'$ de sorte que $X_1\lambda \bowtie_{\Psi'_0\lambda'} X_2\lambda$.

En effet, soit $\forall\beta.M \bowtie N$ dans Ψ'_0. Par l'hypothèse sur Ψ_0, cette équation est totalement stratifiée :

$$\begin{aligned} \mathrm{maxar}(\forall\beta.M \bowtie N) &< \mathrm{maxpar}(\forall\beta.M \bowtie N) \\ &\leq \mathrm{maxpar}((\forall\beta.M \bowtie N)\lambda) \\ &\leq \mathrm{maxpar}(X_2\lambda) \end{aligned}$$

Par conséquent, $X_2\lambda$ ne peut pas être un sous-terme de $\mathrm{var}(\forall\beta.M \bowtie N)\lambda$, d'où

$$(\forall\beta.M \bowtie N)\lambda = (\forall\beta.M \bowtie N)\lambda'.$$

Ainsi, nous avons $X_1\lambda \bowtie_{\Psi'_0\lambda'} X_2\lambda$, avec $\mathrm{maxpar}(\Psi'_0\lambda') \leq \mathrm{maxpar}(X_2\lambda)$. Or, pour tout $X \in \mathrm{dom}(\Sigma)$, sachant que $X\lambda' \bowtie_{\{X_1\lambda \bowtie X_2\lambda\}} X\lambda$, on a donc

$$X\lambda' \bowtie_{\Psi'_0\lambda'} X\lambda.$$

En particulier, sachant que $\Psi'_0 \subseteq \Psi$, pour tout $\forall\beta.M \bowtie N$ dans Ψ (en supposant $\beta \cap \mathrm{supp}(\lambda') = \emptyset$), on a

$$M\lambda \bowtie_{\Psi\lambda'} M\lambda' \bowtie_{\Psi\lambda'} N\lambda' \bowtie_{\Psi\lambda'} N\lambda.$$

Par conséquent, pour tout $X \in \mathcal{Y}$, l'hypothèse $X\theta \bowtie_{\Psi\lambda} X\lambda$ implique

$$X\theta \bowtie_{\Psi\lambda'} X\lambda \bowtie_{\Psi\lambda'} X\lambda'$$

ce qui permet de conclure $\theta, \lambda' \models \Sigma'$.

Pour finir nous vérifions les deux derniers points. Comme λ et λ' coïncident au premier ordre, et $t\lambda$ est $\mathcal{R}$-réduit, on a $\mathrm{t}(\lambda, \Sigma) = \mathrm{t}(\lambda', \Sigma')$. En outre, $\mathrm{c}(\Sigma') = \mathrm{c}(\Sigma)$ et $\mathrm{s}(\Sigma') \leq \mathrm{s}(\Sigma)$. Il suffit donc de vérifier le dernier point : pour tous t_0 $\mathcal{R}$-réduit clos, $\Phi_1 \subseteq \Phi^{\mathrm{S}}$ et $\Psi_1 \subseteq \Phi$, on a

$$\mathrm{R}^*(\lambda', \Phi_1\,\sigma', (\Psi_0 \cup \Psi_1)\,\sigma', t_0) \subseteq \mathrm{R}^*(\lambda, \Phi_1, \Psi_1, t_0)\,\sigma'$$

(En effet, dans le cas où $\mathrm{s}(\Sigma') = \mathrm{s}(\Sigma)$, pour $(\Phi_1, \Psi_1) = (\Phi^{\mathrm{S}}, \Psi)$, on obtient : pour tout t_0,

$$\mathrm{R}^*(\lambda', (\Phi\sigma')^{\mathrm{S}}, \Psi\sigma', t_0) = \mathrm{R}^*(\lambda', \Phi^{\mathrm{S}}\sigma', \Psi\sigma', t_0) \subseteq \mathrm{R}^*(\lambda, \Phi^{\mathrm{S}}, \Psi, t_0)\sigma'$$

et par conséquent $\mathrm{r}(\lambda, \Sigma) \dot{\geq} \mathrm{r}(\lambda', \Sigma')$.)

Pour cela, nous utilisons le lemme 4.33. En effet, sachant que

- $\Psi'_0 \subseteq \Psi_0$ est totalement stratifié,
- $\mathrm{maxpar}(X\lambda') \leq \mathrm{maxpar}(X\lambda)$,
- $X\lambda' \bowtie_{\Psi'_0\lambda'} X\lambda$,
- $X\lambda' \neq X\lambda \Rightarrow \mathrm{maxpar}(\Psi'_0\lambda) \leq \mathrm{maxpar}(X\lambda)$ (en effet d'une part $X\lambda' \neq X\lambda \Rightarrow X_2\lambda \in \mathrm{st}(X\lambda)$, et d'autre part $\mathrm{maxpar}(\Psi'_0\lambda) \leq \mathrm{maxpar}(X_2\lambda)$)

on obtient que pour tout t_0 $\mathcal{R}$-réduit et clos, on a

$$\mathrm{R}^*(\lambda', \Phi_1\,\sigma', (\Psi'_0 \cup \Psi_1)\,\sigma', t_0) \subseteq \mathrm{R}^*(\lambda, \Phi_1, \Psi_1, t_0)\,\sigma'$$

Or $\Psi'_0 \subseteq \Psi_0$ implique

$$\mathrm{R}^*(\lambda', \Phi_1\,\sigma', (\Psi_0 \cup \Psi_1)\,\sigma', t_0) \subseteq \mathrm{R}^*(\lambda', \Phi_1\,\sigma', (\Psi'_0 \cup \Psi_1)\,\sigma', t_0)$$

3. Si $\mathrm{maxpar}(X_1\lambda) > \mathrm{maxpar}(X_2\lambda)$ ou que $X_2\lambda$ est un sous-terme strict de $X_1\lambda$, alors en particulier on a : $\mathrm{maxpar}(X_2\lambda) \leq \mathrm{maxpar}(X_1\lambda) \leq \mathrm{ar}(X_1)$ et $X_1\lambda$ n'est pas un sous-terme strict de $X_2\lambda$. On procède alors comme dans le cas 2. en inversant les rôles de X_1 et X_2 : λ^1 est solution de

$$\begin{cases} Y_1 & =^? & C_1[M_1, \ldots, M_m]\{X_1 \mapsto D_2[M_1, \ldots, M_m]\} \\ & \ldots & \\ Y_n & =^? & C_n[M_1, \ldots, M_m]\{X_1 \mapsto D_2[M_1, \ldots, M_m]\} \end{cases}$$

vérifiant $\lambda^1|_{\mathrm{var}(\Sigma)-\{Y_1,\ldots,Y_n\}} = \lambda|_{\mathrm{var}(\Sigma)-\{Y_1,\ldots,Y_n\}}$ et $\lambda' = \sigma|_{\mathrm{var}(\Sigma')}\{X_1 \mapsto X_2\lambda\}\lambda^1$. □

Lemme 4.37 (**Fail**). *Soit* $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ *un système de contraintes. Supposons qu'il existe* λ *clos de domaine* $\mathrm{var}(\Sigma)$ *vérifiant : pour tout* $(X \rhd^? t_0) \in \mathcal{C}$*, si* $t_0 \in \mathrm{nps}(\mathcal{C})$ *alors*

$$X\lambda \rhd_{\Phi^{\mathrm{S}}\lambda} t\lambda$$

où Φ^{S} *est le sous-ensemble des règles totalement stratifiées de* Φ*.*

Alors la règle **Fail** *ne s'applique pas à* Σ*.*

Démonstration. Supposons que **Fail** s'applique à une contrainte $X \rhd^? t_0$ de $\mathcal{C}$. En particulier $t_0 = f(t_1, \ldots, t_n) \in \mathrm{nps}(\mathcal{C})$ et $f \notin \mathcal{F}_{\mathrm{pub}}$.

Par hypothèse, on a $X\lambda \rhd^?_{\Phi^{\mathrm{S}}\lambda} t_0\lambda$. Sachant que $f \notin \mathcal{F}_{\mathrm{pub}}$, ceci implique qu'il existe $M \rhd t$ dans Φ^{S} tel que $X\lambda = M\lambda$ et $t_0\lambda = t\lambda$.

Par bonne formation de λ, et comme M est totalement stratifié, $X\lambda = M\lambda$ implique $\mathrm{maxar}(M) < \mathrm{maxpar}(M) \leq \mathrm{ar}(X)$. Or, les termes t_0 et t sont unifiables du fait de $t_0\lambda = t\lambda$; ce qui contredit les prémisses de la règle. □

Lemme 4.38 (**Normalize-free-variables**). *Soit* $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ *un système accessible. Supposons les conditions suivantes vérifiées :*

- $\theta, \lambda \models \Sigma$ *;*
- *il existe une variable* x_0 *dans* $\mathrm{var}(\mathcal{C}) - \mathrm{var}(\mathcal{C}_{\rhd})$*, telle que* $x_0\lambda$ *n'est pas en forme normale.*

Alors, $\lambda' = \sigma|_{\mathrm{var}(\Sigma)}\{x_0 \mapsto x_0\lambda{\downarrow_{\mathcal{R}}}\}\lambda$ *vérifie*

- $\theta, \lambda' \models \Sigma$*,*
- $(\lambda, \Sigma) >_c (\lambda', \Sigma)$*.*

Démonstration. Par la propriété de stratification (proposition 4.19), on a $x_0 \notin \mathrm{var}(\Phi)$, d'où $\theta, \lambda' \models \Sigma$. Cependant, $x_0 \in \mathrm{var}(\mathcal{C})$ donc $\mathrm{t}(\lambda, \Sigma) > \mathrm{t}(\lambda', \Sigma')$. □

Lemme 4.39 (**Narrowing**). *Soit* $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ *un système accessible. Supposons les conditions suivantes vérifiées :*

- $\theta, \lambda \models \Sigma$ *;*
- Σ *est pré-résolu ;*
- *il existe un sous-terme* $t_0 = f(t_1, \ldots, t_n)$ *dans* $\mathrm{st}^1(\Phi, \mathcal{C})$ *tel que* $t_0\lambda$ *n'est pas en forme* $\mathcal{R}$*-normale, mais les termes* $t_1\lambda$*, . . . ,* $t_n\lambda$ *le sont.*

Alors, $f(t_1, \ldots, t_n) \notin \mathrm{st}(\mathcal{N})$ *et il existe une règle fraîche* $l \to r$ *issue de* $\mathcal{R}$ *telle que* $t_0\lambda = l\mu_0$ *pour un certain* μ_0 *clos de domaine* $\mathrm{var}(l)$*. Soit* $\mu = \mathrm{mgu}(f(t_1, \ldots, t_n), l)$ *et*

$$\Sigma' = (\ \Phi\{t_0 \mapsto r\};\ \Psi;\ \mathcal{C}\{t_0 \mapsto r\};\ \mathcal{N} \cup \{t_1, \ldots, t_n\};\ \sigma\ )\,\mu.$$

On a $\Sigma \Longrightarrow_{\mathbf{Narrowing}} \Sigma'$ *et il existe* λ' *tel que :*

- $\theta, \lambda' \models \Sigma'$ *;*
- $(\lambda, \Sigma) >_c (\lambda', \Sigma')$*.*

Enfin, si Σ *est totalement stratifié et si l'on note*

$$\begin{aligned} \Phi_0 &= \{(M \rhd t) \in \Phi \mid t_0 \in \mathrm{st}(t)\} \\ k_1 &= \min\{\mathrm{maxpar}(M) \mid (M \rhd t) \in \Phi_0\} \in [0, \infty] \\ \Phi_1 &= \{(M \rhd t) \in \Phi - \Phi_0 \mid \mathrm{maxpar}(M) \leq k_1\}\} \end{aligned}$$

le même λ' *vérifie en outre : pour tout* t_1 $\mathcal{R}$*-réduit clos,*

$$\mathrm{R}^*(\lambda', \Phi_1\sigma', \Psi\sigma', t_1) \subseteq \mathrm{R}^*(\lambda, \Phi_1, \Psi, t_1)\sigma'$$

où $\Phi_1\sigma' = \Phi_1\mu \subseteq \Phi' = (\Phi\{t_0 \mapsto r\})\mu$*.*

Démonstration. Le système Σ étant pré-résolu, l'ensemble des contraintes de déductibilité dans $\mathcal{C}$ est de la forme

$$\mathcal{C}_{\rhd} = \{X_1 \rhd x_1, \ldots, X_n \rhd^? x_n\}$$

où les variables X_1, …, X_n, x_1, …, x_n sont deux à deux distinctes. Comme $\theta, \lambda \models \Sigma$, il existe des règles $M_1 \rhd t_1$, …, $M_m \rhd t_m$, et des contextes publics C_1, …, C_n tels que λ est solution du système d'équations $\mathcal{U}$:

$$\left\{\begin{array}{rcl} X_1 & =^? & C_1[M_1, \ldots, M_m] \\ & \cdots & \\ X_n & =^? & C_n[M_1, \ldots, M_m] \\ x_1 & =^? & C_1[t_1, \ldots, t_m] \\ & \cdots & \\ x_n & =^? & C_n[t_1, \ldots, t_m] \end{array}\right.$$

Notons $i \prec j$ lorsque $x_i \in C_j[t_1, \ldots, t_m]$. Σ étant stratifié (proposition 4.19), d'après le lemme 4.20, $\prec$ est sans cycle, et pour tous $i \prec j$, $\mathrm{maxpar}(X_i\lambda) \leq \mathrm{maxpar}(X_j\lambda)$.

Considérons le système $\mathcal{U}'$:

$$\left\{\begin{array}{rcl} X_1 & =^? & C_1[M_1, \ldots, M_m, M] \\ & \cdots & \\ X_n & =^? & C_n[M_1, \ldots, M_m, M] \\ x_1 & =^? & C_1[t_1\{t_0 \mapsto r\mu_0\}, \ldots, t_m\{t_0 \mapsto r\mu_0\}] \\ & \cdots & \\ x_n & =^? & C_n[t_1\{t_0 \mapsto r\mu_0\}, \ldots, t_m\{t_0 \mapsto r\mu_0\}] \end{array}\right.$$

Comme $\mathrm{var}(t_i\{t_0 \mapsto r\mu_0\}) \subseteq \mathrm{var}(t_i)$ (μ_0 est clos), et comme $\prec$ est sans cycle, la seconde moitié de $\mathcal{U}'$ est encore en forme DAG-résolue. Il existe donc une solution λ' de $\mathcal{U}'$ telle que

- $\mathrm{dom}(\Sigma') = \mathrm{var}(\Sigma') = \mathrm{var}(\Sigma) \uplus \mathrm{var}(l)$
- pour tout $V \in \mathrm{var}(\Sigma) - \{x_1, \ldots, x_n\}$, $V\lambda' = V\lambda$;
- pour tout $x \in \mathrm{var}(l)$, $x\lambda' = x\mu_0$;
- pour tout i, $x_i\lambda \to^*_{\mathcal{R}} x_i\lambda'$; (en particulier, $x_i \in \mathrm{st}(\mathcal{N}) \Rightarrow x_i\lambda = x_i\lambda'$)
- enfin, pour tout i, $x_i\lambda \neq x_i\lambda' \Rightarrow \exists j \prec^* i, \exists w_k \in \mathrm{par}(C_j), t_0 \in \mathrm{st}(t_k)$.

(Les deux dernières conditions se vérifient par récurrence sur $\prec$.)

Par construction, λ' satisfait les contraintes de déductibilité de Σ'. Comme pour tout $x \in \mathrm{var}(\Sigma)$, on $x\lambda' =_E x\lambda$, λ' satisfait aussi les contraintes d'égalité et de différence. Enfin, par la proposition 4.16, les variables $x \in \mathrm{var}(\mathrm{supp}(\sigma)\sigma)$ satisfont $x \in \mathrm{st}(\mathcal{N})$ et par conséquent $x\lambda' = x\lambda$. On en déduit que la condition : pour tout $V \in \mathrm{var}(\Sigma')$, $V\sigma'\lambda' = V\lambda'$ est satisfaite ; d'où $\theta, \lambda' \models \Sigma'$.

De plus, comme t_0 est sous-terme de Φ, de $\mathcal{C}_=$ ou de $\mathcal{C}_{\neq}$, $\mathrm{t}(\lambda, \Sigma) > \mathrm{t}(\lambda', \Sigma')$; d'où $(\lambda, \Sigma) >_c (\lambda', \Sigma')$.

Enfin, vérifions la dernière condition dans le cas où Σ est totalement stratifié. Pour cela, on vérifie que pour tout $x \in \mathrm{var}(\Phi_1)$, $x\lambda' = x\lambda$. En effet, le lemme 4.33 s'applique ensuite à $\Phi_1 = \Phi_1^{\mathrm{S}}$, $\Psi_1 = \Psi$ et $\Psi_0 = \emptyset$.

Par contradiction, supposons qu'il existe $x_i \in \mathrm{var}(\Phi_1)$ tel que $x_i\lambda' \neq x_i\lambda$. Avec les notations précédentes, ceci implique qu'il existe $j \prec^* i$ et $(M_k \rhd t_k) \in \Phi_0$

tel que $w_k \in \mathrm{par}(C_j)$. En particulier, $\mathrm{maxpar}(X_j\lambda) \leq \mathrm{maxpar}(X_i\lambda)$. Comme $M_k\lambda$ est sous-terme de $X_j\lambda = C_j[M_1, \ldots, M_n]\lambda$, on en déduit

$$\mathrm{ar}(X_i) \geq \mathrm{maxpar}(X_j\lambda) \geq \mathrm{maxpar}(M_k\lambda) \geq \mathrm{maxpar}(M_k).$$

D'autre part, Σ étant pré-résolu et stratifié, $x_i \in \mathrm{var}(\Phi_1)$ implique $X_i \preceq M$ pour un certain $(M \rhd t) \in \Phi_1$. Sachant que $\mathrm{maxpar}(M) \leq k_1$ par définition de Φ_1, et que $\mathrm{maxar}(M) < \mathrm{maxpar}(M)$ (car Σ est totalement stratifié), on obtient finalement $\mathrm{ar}(X_i) < \mathrm{maxpar}(M) \leq k_1 \leq \mathrm{maxpar}(M_k)$; contradiction. □

Lemme 4.40 (**Constrain**). *Soit Σ un système accessible. Supposons les conditions suivantes vérifiées :*
- $\theta, \lambda \models \Sigma$,
- $\Sigma = \Phi;\ \Psi;\ \mathcal{C} \uplus \{t =^?_E t'\};\ \mathcal{N};\ \sigma$,
- *les termes $t\lambda$ et $t'\lambda$ en forme $\mathcal{R}$-normale.*

Alors $t\lambda = t'\lambda$. Soit $\mu = \mathrm{mgu}(t, t')$ et

$$\Sigma' = (\ \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N} \cup \{t\};\ \sigma\ )\,\mu.$$

On a $\Sigma \Longrightarrow_{\mathbf{Constrain}} \Sigma'$ et de plus $\lambda' = \lambda$ vérifie
- $\theta, \lambda' \models \Sigma'$,
- $(\lambda, \Sigma) >_c (\lambda', \Sigma')$, *et*
- *pour tout t_0 $\mathcal{R}$-réduit clos,* $\mathrm{R}(\lambda', \Sigma', t_0) \subseteq \mathrm{R}(\lambda, \Sigma, t_0)$.

Démonstration. $\theta, \lambda' \models \Sigma'$ découle de $\mu|_{\mathrm{var}(\Sigma)}\lambda = \lambda$ et $t\lambda$ normal. On a bien $\mathrm{t}(\lambda, \Sigma) = \mathrm{t}(\lambda', \Sigma')$, $\mathrm{s}(\Sigma) \geq \mathrm{s}(\Sigma')$ et $\mathrm{c}(\Sigma') = \mathrm{c}(\Sigma) - 1$.

Enfin, la dernière propriété (et donc $\mathrm{r}(\lambda, \Sigma) \geq \mathrm{r}(\lambda', \Sigma')$) découle du lemme 4.33 de la même manière que pour la règle **Project**. □

Lemme 4.41 (**Context-{1,2}**). *Soit Σ un système accessible tel que les conditions suivantes sont vérifiées :*
- $\theta, \lambda \models \Sigma$;
- $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ *est pré-résolu et totalement stratifié ;*
- *pour tout t $\mathcal{R}$-réduit clos,* $\mathrm{R}(\lambda, \Sigma, t) = \emptyset$;
- *il existe une contrainte $X_0 \rhd^? x_0$ dans $\mathcal{C}$, un contexte public C_0, des règles $M_1 \rhd t_1$, …, $M_m \rhd t_m$ dans Φ, et une position p dans C_0 tels que*
  - $X_0\lambda = C_0[M_1, \ldots, M_m]\lambda$;
  - $x_0\lambda = C_0[t_1, \ldots, t_m]\lambda$;
  - *le contexte public $C = C_0|_p$ est de la forme $C = f(C'_1, \ldots, C'_k)$;*
  - *le terme $C[t_1, \ldots, t_m]\lambda$ n'est pas en forme $\mathcal{R}$-normale, mais les termes $C'_1[t_1, \ldots, t_m]\lambda$, …, $C'_k[t_1, \ldots, t_m]\lambda$ le sont.*

Alors, il existe une règle fraîche $l \to r$ issue de $\mathcal{R}$ telle que

$$C[t_1, \ldots, t_m]\lambda = l\mu_0 \tag{4.4.1}$$

pour un certain μ_0 clos de domaine $\mathrm{var}(l)$.

Soit $l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$ (un renommage frais de) la décomposition donnée par le lemme 4.6 à partir de l'équation 4.4.1, en particulier : comme les $t_1\lambda$, …, $t_m\lambda$ sont $\mathcal{R}$-réduits, D est propre.

Pour tout $1 \leq i \leq n$, il existe $1 \leq j_i \leq m$ tel que $t_{j_i}\lambda = l_i\mu_0$ et tel que les propriétés qui suivent sont vérifiées.

Il existe un unificateur plus général $\mu = \mathrm{mgu}(\,(t_{j_1}, \ldots, t_{j_n}), (l_1, \ldots, l_n)\,)$. *Soit* $Y_1, \ldots, Y_p$, $Z_1, \ldots, Z_q$ *des variables fraîches d'arité* $\mathrm{ar}(X_0)$.

Nous distinguons deux cas.

(1) S'il existe D' *tel que* $r = D'[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$, *alors posons*

$$\begin{array}{rcl} M & = & D[M_{j_1}, \ldots, M_{j_n}, Y_1, \ldots, Y_p, Z_1, \ldots, Z_q] \\ M' & = & D'[M_{j_1}, \ldots, M_{j_n}, Y_1, \ldots, Y_p, Z_1, \ldots, Z_q] \\ \Sigma' & = & (\Phi;\, \Psi \cup \{\forall Z_1, \ldots, Z_q.M \bowtie M'\}; \\ & & \qquad \mathcal{C} \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\};\, \mathcal{N} \cup \{l_1, \ldots, l_n\};\, \sigma)\mu \end{array}$$

de sorte que $\Sigma \Longrightarrow_{\mathbf{Context-2}} \Sigma'$.

(2) Sinon, $\mathcal{R}$ *ayant la propriété sous-terme, on a* $r \in \mathrm{st}_{\neq}(l_1, \ldots, l_n) \cup \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}}$. *Pour tout* $1 \leq j \leq q$, *soit* $\mathsf{c}_j = \mathsf{c}_{\mathrm{sort}(Z_j)}$ *la constante fixée précédemment de même sorte que* Z_j. *Posons*

$$\begin{array}{rcl} M & = & D[M_{j_1}, \ldots, M_{j_n}, Y_1, \ldots, Y_p, Z_1, \ldots, Z_q] \\ M' & = & D[M_{j_1}, \ldots, M_{j_n}, Y_1, \ldots, Y_p, \mathsf{c}_1, \ldots, \mathsf{c}_q] \\ \Sigma' & = & (\Phi \cup \{M' \rhd r\};\, \Psi \cup \{\forall Z_1, \ldots, Z_q.M \bowtie M'\}; \\ & & \qquad \mathcal{C} \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\};\, \mathcal{N} \cup \{l_1, \ldots, l_n\};\, \sigma)\mu \end{array}$$

de sorte que $\Sigma \Longrightarrow_{\mathbf{Context-1}} \Sigma'$.

Dans les deux cas, il existe λ' *tel que :*

- $\theta, \lambda' \models \Sigma'$;
- $(\lambda, \Sigma) >_c (\lambda', \Sigma')$;
- *étant donné* $\Phi_1 \triangleq \{(M \rhd t) \in \Phi \mid \mathrm{maxpar}(M) \leq \mathrm{ar}(X_0)\}$, *pour tout* t $\mathcal{R}$*-réduit clos, on a* $\mathrm{R}^*(\lambda', \Phi_1\mu, \Psi, t) = \emptyset$.

Démonstration. Le système Σ étant pré-résolu, l'ensemble des contraintes de déductibilité dans $\mathcal{C}$ est de la forme

$$\mathcal{C}_{\rhd} = \{X_0 \rhd^? x_0, X_1 \rhd x_1, \ldots, X_k \rhd^? x_k\}$$

où les variables X_0, X_1, …, X_k, x_0, x_1, …, x_k sont deux à deux distinctes. Comme $\theta, \lambda \models \Sigma$, quitte à augmenter m et les règles $M_1 \rhd t_1$, …, $M_m \rhd t_m$, on peut supposer qu'il existe des contextes publics $C_1, \ldots, C_k$ tels que λ est solution du système d'équations $\mathcal{U}$:

$$\left\{\begin{array}{rcl} X_0 & =^? & C_0[M_1, \ldots, M_m] \\ X_1 & =^? & C_1[M_1, \ldots, M_m] \\ & \ldots & \\ X_k & =^? & C_k[M_1, \ldots, M_m] \\ x_0 & =^? & C_0[t_1, \ldots, t_m] \\ x_1 & =^? & C_1[t_1, \ldots, t_m] \\ & \ldots & \\ x_k & =^? & C_k[t_1, \ldots, t_m] \end{array}\right.$$

Avant tout, notons que Σ étant stratifié et pré-résolu, pour tous i et j tels que $x_i \in \mathrm{var}(C_j[t_1, \ldots, t_m])$, on a $X_i \preceq C_j[M_1, \ldots, M_m]$; d'où par stratification totale (et bonne formation de λ) :

$$\mathrm{ar}(X_i) < \mathrm{maxpar}(C_j[M_1, \ldots, M_m]) \leq \mathrm{ar}(X_j) \tag{4.4.2}$$

Pour les mêmes raisons, ces inégalités sont vraies a fortiori lorsque $X_i \in \mathrm{var}(C_j[M_1, \ldots, M_m])$.

Par ailleurs, d'après le lemme 4.6, la décomposition propre

$$l = D[l_1, \ldots, l_n, y_1, \ldots y_p, z_1, \ldots, z_q]$$

est telle qu'il existe
- des paramètres $\alpha_1, \ldots, \alpha_{m_0+m_1} \in \mathrm{par}(D) = \{w_1^{\tau_1}, \ldots, w_{n+p+q}^{\tau_{n+p+q}}\}$,
- des paramètres $\alpha'_1, \ldots, \alpha'_{m_0} \in \mathrm{par}(C)$, et
- des contextes publics $D_0, \ldots, D_{m_1}$

tels que

(i) $D = D_0[\alpha_1, \ldots, \alpha_{m_0+m_1}]$ avec

$$\begin{array}{rcl} \{\alpha_1, \ldots, \alpha_{m_0}\} & = & \{w_1^{\tau_1}, \ldots, w_n^{\tau_n}\} \\ \{\alpha_{m_0+1}, \ldots, \alpha_{m_0+m_1}\} & = & \{w_{n+1}^{\tau_{n+1}}, \ldots, w_{n+p+q}^{\tau_{n+p+q}}\} \end{array}$$

(ii) $C = C_0|_p = D_0[\alpha'_1, \ldots, \alpha'_{m_0}, D_1, \ldots, D_{m_1}]$.

Pour commencer nous cherchons à remplacer dans le système précédent $C = C_0|_p$ par un contexte public C'' de la forme

$$C'' = D[\alpha''_1, \ldots, \alpha''_n, D'_1, \ldots, D'_{p+q}]$$

En général, C n'est pas présentable sous cette forme car il est possible que $1 \leq k < k' \leq m_0 + m_1$ et pourtant $\alpha_k = \alpha_{k'}$.

Pour chaque $1 \leq i \leq n$, parmi l'ensemble (non vide, par (1))

$$\{k' \mid 1 \leq k' \leq m_0,\ \alpha_{k'} = w_i^{\tau_i}\}$$

on choisit k'_i de sorte que $\mathrm{maxpar}(\alpha'_{k'_i}[M_1, \ldots, M_m])$ soit minimal, et l'on pose $\alpha'_{k'_i}[M_1, \ldots, M_m] = M_{j_i}$ où $j_i \in \{1, \ldots, m\}$.

De même, pour chaque $n+1 \leq i \leq n+p+q$, parmi l'ensemble (non vide)

$$\{k \mid 1 \leq k \leq m_1,\ \alpha_{m_0+k} = w_i^{\tau_i}\}$$

on choisit k_i de sorte que $\mathrm{maxpar}(D_{k_i}[M_1, \ldots, M_m])$ soit minimal.

Soit

$$C'' = D[\, w_{j_1}, \ldots, w_{j_n}, D_{k_{n+1}}, \ldots, D_{k_{n+p+q}}\,]$$

Par construction, on a $\mathrm{maxpar}(C''[M_1, \ldots, M_m]) \leq \mathrm{maxpar}(C[M_1, \ldots, M_m])$ et de plus

$$\begin{array}{rcl} C''[t_1, \ldots, t_m]\lambda & = & D[\, t_{j_1}\lambda, \ldots, t_{j_n}\lambda, \\ & & \qquad D_{k_{n+1}}[t_1, \ldots, t_m]\lambda, \ldots, D_{k_{n+p+q}}[t_1, \ldots, t_m]\lambda\,] \\ & = & D[\, l_1\mu_0, \ldots, l_n\mu_0,\ y_1\mu_0, \ldots, y_p\mu_0,\ z_1\mu_0, \ldots, z_q\mu_0\,] \\ & = & l\mu_0 \\ & = & C[t_1, \ldots, t_m]\lambda \end{array}$$

Du dernier point, sachant que pour tout t clos $\mathcal{R}$-réduit, $\mathrm{R}(\lambda, \Sigma, t) = \mathrm{R}(\lambda, \Phi^{\mathrm{S}}, \Psi, t) = \emptyset$ avec ici $\Phi^{\mathrm{S}} = \Phi$, on déduit de la même manière que pour la preuve de **Coalesce** qu'il existe $\Psi_0 \subseteq \Psi$ tel que

$$C''[M_1, \ldots, M_m]\lambda \bowtie_{\Psi_0\lambda} C[M_1, \ldots, M_m]\lambda$$

et

$$\begin{array}{rcl} \mathrm{maxpar}(\Psi_0\lambda) & \leq & \mathrm{maxpar}(C[M_1,\ldots,M_m]\lambda, C''[M_1,\ldots,M_m]\lambda) \\ & = & \mathrm{maxpar}(C[M_1,\ldots,M_m]) \end{array}$$

la dernière égalité étant due à l'inégalité précédente $\mathrm{maxpar}(C''[M_1,\ldots,M_m]) \leq \mathrm{maxpar}(C[M_1,\ldots,M_m])$ et au fait que Φ est totalement stratifié. Pour les mêmes raisons, on a également $C''[M_1,\ldots,M_m] \preceq C[M_1,\ldots,M_m]$.

Considérons le système $\mathcal{U}''$:

$$\left\{\begin{array}{rcl} X_0 & =^? & (C_0[C'']_p)[M_1,\ldots,M_m] \\ X_1 & =^? & C_1[M_1,\ldots,M_m] \\ & \ldots & \\ X_k & =^? & C_k[M_1,\ldots,M_m] \\ x_0 & =^? & (C_0[C'']_p)[t_1,\ldots,t_m] \\ x_1 & =^? & C_1[t_1,\ldots,t_m] \\ & \ldots & \\ x_k & =^? & C_k[t_1,\ldots,t_m] \end{array}\right.$$

Étant donné ce qui précède et sachant que les M_1, …, M_m ne sont pas des variables (proposition 4.19), le lemme 4.18 s'applique à $\mathcal{U}$ et λ pour donner l'existence de λ'' tel que

- $\mathrm{dom}(\lambda'') = \mathrm{dom}(\lambda)$;
- λ'' est solution de $\mathcal{U}''$;
- pour tout $V \in \mathrm{dom}(\lambda) - \{X_0,\ldots,X_k\}$, $V\lambda'' = V\lambda$;
- pour tout X_i,
  - $X_i\lambda'' \bowtie_{(C[M_1,\ldots,M_m]\bowtie C''[M_1,\ldots,M_m])\lambda} X_i\lambda$, soit encore $X_i\lambda'' \bowtie_{\Psi_0\lambda} X_i\lambda$,
  - $\mathrm{maxpar}(X_i\lambda'') \leq \mathrm{maxpar}(X_i\lambda)$, et
  - $X_i\lambda'' \neq X_i\lambda \Rightarrow C[M_1,\ldots,M_m]\lambda \in \mathrm{st}(X_i\lambda)$.

En particulier, sachant que pour tout $X \in \mathrm{var}(\Psi_0)$, $\mathrm{ar}(X) < \mathrm{maxpar}(\Psi_0) \leq \mathrm{maxpar}(C[M_1,\ldots,M_m])$ (par stratification totale), le dernier point implique que pour tout $X \in \mathrm{var}(\Psi_0)$, $X\lambda'' = X\lambda$; d'où $\Psi_0\lambda'' = \Psi_0\lambda$.

De plus, sachant que

- $\mathrm{R}^*(\lambda, \Phi_1, \Psi, t) = \emptyset$
- $\Psi_0 \subseteq \Psi$ est totalement stratifié,
- pour tout $x \in \mathrm{dom}(\lambda'')$, $x\lambda'' = x\lambda$,
- pour tout $X \in \mathrm{dom}(\lambda'')$,
  - $\mathrm{maxpar}(X\lambda'') \leq \mathrm{maxpar}(X\lambda)$,
  - $X\lambda'' \bowtie_{\Psi_0\lambda} X\lambda$,
  - et $X\lambda'' \neq X\lambda \Rightarrow \mathrm{maxpar}(\Psi_0\lambda) \leq \mathrm{maxpar}(X\lambda)$ (en effet d'une part $X\lambda'' \neq X\lambda \Rightarrow X_0\lambda \in \mathrm{st}(X\lambda)$, et d'autre part $\mathrm{maxpar}(\Psi_0\lambda) = \mathrm{maxpar}(\Psi) \leq \mathrm{maxpar}(C[M_1,\ldots,M_m]) \leq \mathrm{maxpar}(X_0\lambda)$) ;

le lemme 4.33 implique que pour tout t $\mathcal{R}$-réduit clos, $\mathrm{R}^*(\lambda'', \Phi_1, \Psi, t) = \emptyset$.

Nous pouvons maintenant aborder les deux cas du théorème.

(1) Si $r = D'[l_1,\ldots,l_n, y_1,\ldots,y_p, z_1,\ldots,z_q]$, alors d'après l'énoncé du théorème, on définit

$$\begin{array}{rcl} M & = & D[M_{j_1},\ldots,M_{j_n}, Y_1,\ldots,Y_p, Z_1,\ldots,Z_q] \\ M' & = & D'[M_{j_1},\ldots,M_{j_n}, Y_1,\ldots,Y_p, Z_1,\ldots,Z_q] \\ \Sigma' & = & (\Phi;\ \Psi \cup \{\forall Z_1,\ldots,Z_q.M \bowtie M'\}; \\ & & \quad \mathcal{C} \cup \{Y_1 \rhd^? y_1,\ldots,Y_p \rhd^? y_p\};\ \mathcal{N} \cup \{l_1,\ldots,l_n\};\ \sigma)\mu \end{array}$$

Comme D utilise tous ces arguments par définition des décompositions, on a $\mathrm{var}(M') \subseteq \mathrm{var}(M)$. Soit de plus

$$C' = D'[\, w_{j_1}, \ldots, w_{j_n}, D_{k_{n+1}}, \ldots, D_{k_{n+p+q}}\,]$$

On a d'une part

$$l\mu_0 = C''[t_1, \ldots, t_m]\lambda \to_{\mathcal{R}} r\mu_0 = C'[t_1, \ldots, t_m]\lambda$$

D'autre part, en posant

$$\begin{array}{rcl} \mu_1 & = & \{Z_i \mapsto D_i[M_1, \ldots, M_m]\}_{n+p+1 \le i \le n+p+q}\, \lambda'' \\ \lambda''' & = & \lambda'' \uplus \{Y_i \mapsto D_i[M_1, \ldots, M_m]\}_{n+1 \le i \le n+p}\, \lambda'', \\ & & \quad y_i \mapsto D_i[t_1, \ldots, t_m]\}_{n+1 \le i \le n+p}\, \lambda'' \} \end{array}$$

sachant que $C''[M_1, \ldots, M_m]\lambda'' = M\mu_1\lambda'''$ et $C'[M_1, \ldots, M_m]\lambda'' = M'\mu_1\lambda'''$, on obtient

$$C''[M_1, \ldots, M_m]\lambda''' \bowtie_{(\forall Z_1, \ldots, Z_q.\, M \bowtie M')\lambda'''} C'[M_1, \ldots, M_m]\lambda'''$$

Par ailleurs, λ''' est solution du système $\mathcal{U}'''$ suivant :

$$\left\{ \begin{array}{lcllcl} X_0 & =^? & (C_0[C'']_p)[M_1, \ldots, M_m] & x_0 & =^? & (C_0[C'']_p)[t_1, \ldots, t_m] \\ X_1 & =^? & C_1[M_1, \ldots, M_m] & x_1 & =^? & C_1[t_1, \ldots, t_m] \\ & \ldots & & & \ldots & \\ X_k & =^? & C_k[M_1, \ldots, M_m] & x_k & =^? & C_k[t_1, \ldots, t_m] \\ Y_1 & =^? & D_{n+1}[M_1, \ldots, M_m] & y_1 & =^? & D_{n+1}[t_1, \ldots, t_m] \\ & \ldots & & & \ldots & \\ Y_p & =^? & D_{n+p}[M_1, \ldots, M_m] & y_p & =^? & D_{n+p}[t_1, \ldots, t_m] \end{array} \right.$$

Considérons le système $\mathcal{U}'$:

$$\left\{ \begin{array}{lcllcl} X_0 & =^? & (C_0[C']_p)[M_1, \ldots, M_m] & x_0 & =^? & (C_0[C']_p)[t_1, \ldots, t_m] \\ X_1 & =^? & C_1[M_1, \ldots, M_m] & x_1 & =^? & C_1[t_1, \ldots, t_m] \\ & \ldots & & & \ldots & \\ X_k & =^? & C_k[M_1, \ldots, M_m] & x_k & =^? & C_k[t_1, \ldots, t_m] \\ Y_1 & =^? & D_{n+1}[M_1, \ldots, M_m] & y_1 & =^? & D_{n+1}[t_1, \ldots, t_m] \\ & \ldots & & & \ldots & \\ Y_p & =^? & D_{n+p}[M_1, \ldots, M_m] & y_p & =^? & D_{n+p}[t_1, \ldots, t_m] \end{array} \right.$$

Sachant que les M_1, …, M_m ne sont pas des variables, que Σ est stratifié, et que les variables Y_i, y_i sont fraîches, en appliquant successivement les lemmes 4.18 et 4.20 au système $\mathcal{U}'''$ et à λ''', on obtient l'existence de λ' tel que

- $\mathrm{dom}(\lambda') = \mathrm{dom}(\lambda''') = \mathrm{dom}(\lambda) \uplus \{Y_1, \ldots, Y_p, y_1, \ldots, y_p\}$;
- λ' est solution de $\mathcal{U}'$;
- pour tout $V \in \mathrm{dom}(\lambda) - \{X_0, \ldots, X_k, Y_1, \ldots, Y_p, x_0, \ldots, x_k, y_1, \ldots, y_p\}$, $V\lambda' = V\lambda'''$;
- pour tout $X \in \{X_0, \ldots, X_k, Y_1, \ldots, Y_p\}$,
  - $X\lambda' \bowtie_{(C''[M_1, \ldots, M_m] \bowtie C'[M_1, \ldots, M_m])\lambda'''} X\lambda'''$, soit encore

$$X\lambda' \bowtie_{(\forall Z_1, \ldots, Z_q.\, M \bowtie M')\lambda'''} X\lambda'''$$

- $\mathrm{maxpar}(X\lambda') \leq \mathrm{maxpar}(X\lambda''')$, et
- $X\lambda' \neq X\lambda''' \Rightarrow C'''[M_1, \ldots, M_m]\lambda''' \in \mathrm{st}(X\lambda''')$; et plus précisément d'après la remarque initiale sur la forme du système $\mathcal{U}$ (inégalité 4.4.2) et la définition de $\mathcal{U}''$, pour tout i, $X_i\lambda'' \neq X_i\lambda$ implique $X_i = X_0$ ou $\mathrm{ar}(X_0) < \mathrm{ar}(X_i)$.

– pour tout $x \in \mathrm{dom}(\lambda')$,
- $x\lambda''' \to_{\mathcal{R}}^{*} x\lambda'$;
- à nouveau, d'après la remarque initiale sur la forme de $\mathcal{U}$ et la définition de $\mathcal{U}''$, $x_i\lambda'' \neq x_i\lambda$ implique $X_i = X_0$ ou $\mathrm{ar}(X_0) < \mathrm{ar}(X_i)$.

Comme M n'est pas une variable (D étant propre) et que $\mathrm{var}(M') \subseteq \mathrm{var}(M)$, le terme $C'''[M_1, \ldots, M_m]\lambda''' = M\mu_1\lambda'''$ n'est pas un sous-terme de $\mathrm{var}(\forall Z_1, \ldots, Z_q.M \bowtie M')\lambda'''$. On a donc

$$(\forall Z_1, \ldots, Z_q.M \bowtie M')\lambda''' = (\forall Z_1, \ldots, Z_q.M \bowtie M')\lambda'$$

Par conséquent, pour tout $X \in \mathcal{Y}$,

$$\begin{array}{lll} X\lambda' & \bowtie_{\{\forall Z_1,\ldots,Z_q.M\bowtie M'\}\lambda'} & X\lambda''' \\ & = & X\lambda'' \\ & \bowtie_{\Psi_0\lambda'\cup\{\forall Z_1,\ldots,Z_q.M\bowtie M'\}\lambda'} & X\lambda \\ & \bowtie_{\Psi\lambda'\cup\{\forall Z_1,\ldots,Z_q.M\bowtie M'\}\lambda'} & X\theta \end{array}$$

On en déduit $\theta, \lambda' \models \Sigma'$ et $(\lambda, \Sigma) >_c (\lambda, \Sigma')$ par un raisonnement similaire au cas de la règle **Narrowing** (lemme 4.39).

De plus, les précisions concernant la forme de $\mathcal{U}$ impliquent $\Phi_1\lambda' = \Phi_1\lambda'''$ par définition de $\Phi_1 = \Phi_1^{\mathrm{S}}$. De même pour toute équation $\forall\beta.N \bowtie N'$ dans Ψ (par hypothèse totalement stratifiée) telle que $(\mathrm{maxpar}(\forall\beta.N \bowtie N')\lambda') < \mathrm{ar}(X_0)$, on a $(\forall\beta.N \bowtie N')\lambda' = \forall\beta.N \bowtie N')\lambda''$. Ainsi, pour tout t $\mathcal{R}$-réduit clos, $\mathrm{R}^*(\lambda'', \Phi_1, \Psi, t) = \emptyset$ implique $\mathrm{R}^*(\lambda', \Phi_1, \Psi, t) = \emptyset$.

Par le lemme 4.32 et dans la mesure $\Phi_1\mu\lambda' = \Phi_1\sigma'\lambda' = \Phi_1\lambda'$, on en déduit finalement

$$\mathrm{R}^*(\lambda', \Phi_1\mu, \Psi, t) = \emptyset$$

(2) Si $r \in \mathrm{st}_{\neq}(l_1, \ldots, l_n)$, alors d'après l'énoncé du théorème, on définit

$$\begin{array}{rcl} M & = & D[M_{j_1}, \ldots, M_{j_n}, Y_1, \ldots, Y_p, Z_1, \ldots, Z_q] \\ M' & = & D[M_{j_1}, \ldots, M_{j_n}, Y_1, \ldots, Y_p, \mathsf{c}_1, \ldots, \mathsf{c}_q] \\ \Sigma' & = & (\Phi \cup \{M' \rhd r\};\ \Psi \cup \{\forall Z_1, \ldots, Z_q.M \bowtie M'\}; \\ & & \quad \mathcal{C} \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\};\ \mathcal{N} \cup \{l_1, \ldots, l_n\};\ \sigma)\mu \end{array}$$

Soit de plus

$$C' = D[\, w_{j_1}, \ldots, w_{j_n}, D_{k_{n+1}}, \ldots, D_{k_{n+p}}, \mathsf{c}_1, \ldots, \mathsf{c}_q \,]$$

D'une part, sachant $\mathrm{var}(r) \cap \{z_1, \ldots, z_q\} = \emptyset$, on a

$$C'[t_1, \ldots, t_m]\lambda \to_{\mathcal{R}} r\mu_0$$

D'autre part, en posant

$$\begin{array}{rcl} \mu_1 & = & \{Z_i \mapsto D_i[M_1, \ldots, M_m]\}_{n+p+1\leq i\leq n+p+q}\,\lambda'' \\ \lambda''' & = & \lambda'' \uplus \{Y_i \mapsto D_i[M_1, \ldots, M_m]\}_{n+1\leq i\leq n+p}\,\lambda'', \\ & & \quad y_i \mapsto D_i[t_1, \ldots, t_m]\}_{n+1\leq i\leq n+p}\,\lambda''\} \end{array}$$

sachant que $C''[M_1,\ldots,M_m]\lambda'' = M\mu_1\lambda'''$ et $C'[M_1,\ldots,M_m]\lambda'' = M'\mu_1\lambda'''$, on obtient

$$C''[M_1,\ldots,M_m]\lambda''' \bowtie_{(\forall Z_1,\ldots,Z_q.M\bowtie M')\lambda'''} C'[M_1,\ldots,M_m]\lambda'''$$

Par ailleurs, λ''' est solution du système $\mathcal{U}'''$ suivant :

$$\left\{\begin{array}{llll} X_0 &=^? & (C_0[C'']_p)[M_1,\ldots,M_m] & x_0 =^? (C_0[C'']_p)[t_1,\ldots,t_m] \\ X_1 &=^? & C_1[M_1,\ldots,M_m] & x_1 =^? C_1[t_1,\ldots,t_m] \\ & \ldots & & \ldots \\ X_k &=^? & C_k[M_1,\ldots,M_m] & x_k =^? C_k[t_1,\ldots,t_m] \\ Y_1 &=^? & D_{n+1}[M_1,\ldots,M_m] & y_1 =^? D_{n+1}[t_1,\ldots,t_m] \\ & \ldots & & \ldots \\ Y_p &=^? & D_{n+p}[M_1,\ldots,M_m] & y_p =^? D_{n+p}[t_1,\ldots,t_m] \end{array}\right.$$

On conclut la preuve de la même manière que le cas précédent en considérant le système $\mathcal{U}'$ (les deux cas ne différant que par la définition de M') :

$$\left\{\begin{array}{llll} X_0 &=^? & C'_0[M_1,\ldots,M_m,M'] & x_0 =^? C'_0[t_1,\ldots,t_m,r\mu_0] \\ X_1 &=^? & C_1[M_1,\ldots,M_m] & x_1 =^? C_1[t_1,\ldots,t_m] \\ & \ldots & & \ldots \\ X_k &=^? & C_k[M_1,\ldots,M_m] & x_k =^? C_k[t_1,\ldots,t_m] \\ Y_1 &=^? & D_{n+1}[M_1,\ldots,M_m] & y_1 =^? D_{n+1}[t_1,\ldots,t_m] \\ & \ldots & & \ldots \\ Y_p &=^? & D_{n+p}[M_1,\ldots,M_m] & y_p =^? D_{n+p}[t_1,\ldots,t_m] \end{array}\right.$$

où $C'_0 = C_0[w_{m+1}]_p$. □

Lemme 4.42 (Relate). *Soit Σ un système accessible. Supposons les conditions suivantes vérifiées :*

- *$\theta,\lambda \models \Sigma$;*
- *$\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$;*
- *il existe des termes M, t, N et s tels que $t\lambda$ est en forme normale, et tels que $(M,t,N,s) \in \mathrm{R}(\lambda,\Sigma,t\lambda)$ — autrement dit : $(M \rhd t) \in \Phi^{\mathrm{S}}$, $N \rhd_{\Phi^{\mathrm{S}}} s$, $s\lambda = t\lambda$ et il n'existe pas de sous-ensemble $\Psi_0 \subseteq \Psi$ tel que*

$$M\lambda \bowtie_{\Psi_0\lambda} N\lambda \qquad \mathrm{maxpar}(\Psi_0\lambda) \leq \mathrm{maxpar}(M\lambda, N\lambda)$$

Soit X une variable fraîche d'arité k_0, et

$$\Sigma' = \Phi;\ \Psi \cup \{X \bowtie M\};\ \mathcal{C} \cup \{X \rhd^? t\};\ \mathcal{N} \cup \{t\};\ \sigma.$$

On a $\Sigma \Longrightarrow_{\mathbf{Relate}} \Sigma'$ et de plus $\lambda' = \lambda \uplus \{X \mapsto N\lambda\}$ vérifie

- *$\theta,\lambda' \models \Sigma'$;*
- *$(\lambda,\Sigma) >_c (\lambda',\Sigma')$;*
- *pour tout t_0 $\mathcal{R}$-réduit clos, $\mathrm{R}^*(\lambda',\Phi^{\mathrm{S}},\Psi,t_0) \subseteq \mathrm{R}^*(\lambda,\Phi^{\mathrm{S}},\Psi,t_0)$.*

Démonstration. Par construction, on a bien $\theta,\lambda' \models \Sigma'$.

De plus, $\mathrm{t}(\lambda',\Sigma') = \mathrm{t}(\lambda,\Sigma)$, $\mathrm{s}(\Sigma') = \mathrm{s}(\Sigma)$. Pour tout t_0 $\mathcal{R}$-réduit clos, on a clairement $\mathrm{R}^*(\lambda',\Phi^{\mathrm{S}},\Psi,t_0) \subseteq \mathrm{R}^*(\lambda,\Phi^{\mathrm{S}},\Psi,t_0)$; d'où en en particulier $\mathrm{R}(\lambda',\Sigma',t_0) \subseteq \mathrm{R}(\lambda,\Sigma,t_0)$. Or, $\mathrm{R}(\lambda',\Sigma',t\lambda) \subseteq \mathrm{R}(\lambda,\Sigma,t\lambda) - \{(M,t,N,s)\}$. On en déduit $\mathrm{r}(\lambda,\Sigma) \dot{>} \mathrm{r}(\lambda',\Sigma')$ et finalement $(\lambda,\Sigma) >_c (\lambda',\Sigma')$. □

Lemme 4.43 (Discard). *Soit Σ un système accessible. Supposons les conditions suivantes vérifiées :*

- $\theta, \lambda \models \Sigma$;
- $\Sigma = \Phi \uplus \{M \rhd t\}; \Psi; \mathcal{C}; \mathcal{N}; \sigma$;
- *il existe un terme N tel que $N \rhd_{\Phi,\mathcal{C}} t$ et $N \preceq M$.*

Soit $\Sigma' = \Phi; \Psi \cup \{N \bowtie M\}; \mathcal{C}; \mathcal{N}; \sigma$. On a $\Sigma \Longrightarrow_{\mathbf{Discard}} \Sigma'$ et il existe λ' tel que

- $\theta, \lambda' \models \Sigma'$,
- *pour tout $x \in \mathrm{var}(\Sigma) = \mathrm{var}(\Sigma')$, $x\lambda' = x\lambda$,*
- $(\lambda, \Sigma) \geq_c (\lambda', \Sigma')$, *et*
- *pour tout t_0 $\mathcal{R}$-réduit clos, $\mathrm{R}(\lambda', \Sigma', t_0) \subseteq \mathrm{R}(\lambda, \Sigma, t_0)$.*

Démonstration. Posons

$$\mathcal{C}_{\rhd} = \{X_1 \rhd s_1, \ldots, X_n \rhd^? s_n\}$$

où les variables $X_1, \ldots, X_n$ sont deux à deux distinctes. Comme $\theta, \lambda \models \Sigma$, il existe des règles $M_1 \rhd t_1, \ldots, M_m \rhd t_m$, et des contextes publics $C_1, \ldots, C_n$ tels que λ est solution du système d'équations $\mathcal{U}$:

$$\left\{\begin{array}{rcl} X_1 & =^? & C_1[M_1, \ldots, M_m, M] \\ & \cdots & \\ X_n & =^? & C_n[M_1, \ldots, M_m, M] \\ s_1 & =^? & C_1[t_1, \ldots, t_m, t] \\ & \cdots & \\ s_n & =^? & C_n[t_1, \ldots, t_m, t] \end{array}\right.$$

Considérons le système $\mathcal{U}_2'$:

$$\left\{\begin{array}{rcl} X_1 & =^? & C_1[M_1, \ldots, M_m, N] \\ & \cdots & \\ X_n & =^? & C_n[M_1, \ldots, M_m, N] \end{array}\right.$$

Du fait que $N \preceq M$ et que les termes M, M_1, …, M_m ne sont pas des variables (proposition 4.19), le lemme 4.18 implique l'existence d'un λ' de domaine $\mathrm{var}(\Sigma') = \mathrm{var}(\Sigma)$, qui soit solution de $\mathcal{U}_2'$ et tel que

- pour tout $V \in \mathrm{var}(\Sigma) - \mathrm{var}(\mathcal{U}_2')$, $V\lambda' = V\lambda$;
- $M\lambda' = M\lambda$;
- $N\lambda' = N\lambda$;
- pour tout $X \in \mathrm{var}(\mathcal{U}_2')$, on a $X\lambda' \bowtie_{\{M\lambda' \bowtie N\lambda'\}} X\lambda$, $\mathrm{maxpar}(X\lambda') \leq \mathrm{maxpar}(X\lambda)$ et de plus $X\lambda' \neq X\lambda \Rightarrow M\lambda \in \mathrm{st}(X\lambda)$.

Soit $\Phi' = \Psi \cup \{M \bowtie N\}$. On en déduit $\theta \bowtie_{\Psi'\lambda'} \lambda'|_{\mathcal{Y}}$, d'où $\theta, \lambda' \models \Sigma'$ (les autres contraintes étant satisfaites par construction).

Par ailleurs, on a bien $\mathrm{t}(\lambda', \Sigma') \leq \mathrm{t}(\lambda, \Sigma)$, $\mathrm{c}(\Sigma') = \mathrm{c}(\Sigma)$ et $\mathrm{s}(\Sigma') \leq \mathrm{s}(\Sigma)$. Montrons que pour tout t_0, $\mathrm{R}(\lambda', \Sigma', t_0) \subseteq \mathrm{R}(\lambda, \Sigma, t_0)$.

En effet, posons $\Psi_0 = \{M \bowtie N\}$. Sachant que pour tout $X \in \mathrm{dom}(\lambda) = \mathrm{dom}(\lambda')$,

- $X\lambda' \bowtie_{\Psi_0\lambda'} X\lambda$,
- $\mathrm{maxpar}(X\lambda') \leq \mathrm{maxpar}(X\lambda)$, et
- $X\lambda' \neq X\lambda \Rightarrow \mathrm{maxpar}(\Psi_0\lambda) = \mathrm{maxpar}(M\lambda) \leq \mathrm{maxpar}(X\lambda)$,

le lemme 4.33 s'applique à Φ^{S}, Ψ et $\sigma' = \sigma$ pour donner : pour tout t_0 $\mathcal{R}$-réduit clos,

$$\begin{aligned} \mathrm{R}(\lambda', \Sigma', t_0) &= \mathrm{R}^*(\lambda', \Phi^{\mathrm{S}}, (\Psi_0 \cup \Psi), t_0) \\ &\subseteq \mathrm{R}^*(\lambda, \Phi^{\mathrm{S}}, \Psi, t_0) \\ &\subseteq \mathrm{R}^*(\lambda, (\Phi \cup \{M \rhd t\})^{\mathrm{S}}, \Psi, t_0) \\ &= \mathrm{R}(\lambda, \Sigma, t_0) \end{aligned}$$ □

4.4.3 Progression

Nous utilisons maintenant les résultats des sous-sections précédentes afin de conclure la preuve de complétude proprement dite.

Lemme 4.44. *Soit Σ_0 un système initial et $\Sigma_0 \Longrightarrow^* \Sigma$ une dérivation respectant les priorités entre règles tels que le système Σ est saturé pour les règles prioritaires (à savoir* **Project**, **Imitate**, **Coalesce** *et* **Discard**).

Si Σ n'est pas résolu et $\theta, \lambda \models \Sigma$, alors il existe λ' et Σ' tels que
- *$\theta, \lambda' \models \Sigma'$,*
- *$\Sigma \Longrightarrow^* \Sigma'$ par une dérivation respectant les priorités entre règles,*
- *Σ' est saturé pour les règles prioritaires, et*
- *$(\lambda, \Sigma) >_c (\lambda', \Sigma')$.*

Démonstration. Posons $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$. D'après le lemme 4.8 et les propositions 4.28 et 4.29, le système Σ étant saturé pour les règles prioritaires, il est pré-résolu et totalement stratifié.

De plus, s'il existe une règle $M \rhd x$ dans Φ, alors l'invariant de stratification (proposition 4.19) implique l'existence d'une contrainte $X \rhd^? x$ dans $\mathcal{C}$ telle que $X \preceq M$. Par conséquent, la règle **Discard** s'applique à $M \rhd x$; contradiction. Par conséquent, on a la propriété :

(a) pour tout $M \rhd t$ dans Φ, t n'est pas une variable.

Nous distinguons plusieurs cas.

1. S'il existe un t clos $\mathcal{R}$-réduit tel que $\mathrm{R}(\lambda, \Sigma, t) \neq \emptyset$, alors considérons un tel t_0 minimal pour l'ordre sous-terme. Soit $(M_1, t_1, M_2, t_2) \in \mathrm{R}(\lambda, \Sigma, t_0)$. Par définition de $\mathrm{R}(\lambda, \Sigma, t_0)$, sachant que $\Phi^{\mathrm{S}} = \Phi$, on a
- $t_1\lambda = t_2\lambda = t_0$,
- $(M_1 \rhd t_1) \in \Phi$,
- $M_2 \rhd_\Phi t_2$, et
- il n'existe pas de $\Psi_0 \subseteq \Psi$ tel que $\mathrm{maxpar}(\Psi_0\lambda) \leq \mathrm{maxpar}(M_1\lambda, M_2\lambda)$ et $M_1\lambda \bowtie_{\Psi_0\lambda} M_2\lambda$.

Compte tenu des points précédents, le lemme de complétude de la règle **Relate** (lemme 4.42) s'applique pour donner λ_1 et $\Sigma_1 = \Phi; \Psi \cup \{X_1 \bowtie M_1\}; \mathcal{C} \cup \{X_1 \rhd^? t_1\}; \mathcal{N}; \sigma$ tels que
- $\Sigma \Longrightarrow_{\mathbf{Relate}} \Sigma_1$,
- $\theta, \lambda_1 \models \Sigma_1$,
- $(\lambda, \Sigma) >_c (\lambda_1, \Sigma_1)$, et
- pour tout t clos $\mathcal{R}$-réduit, $\mathrm{R}^*(\lambda_1, \Phi^{\mathrm{S}}, \Psi, t) \subseteq \mathrm{R}^*(\lambda, \Phi^{\mathrm{S}}, \Psi, t)$.

En particulier, le terme $t_0 = t_1\lambda$ étant minimal, et Φ étant totalement stratifié, on a pour tout $t \in \mathrm{st}_{\neq}(t_0)$,

$$\mathrm{R}^*(\lambda_1, \Phi, \Psi, t) = \mathrm{R}^*(\lambda_1, \Phi^{\mathrm{S}}, \Psi, t) \subseteq \mathrm{R}^*(\lambda, \Phi^{\mathrm{S}}, \Psi, t) = \mathrm{R}(\lambda, \Sigma, t) = \emptyset \qquad (4.4.3)$$

Or, pour tout successeur (λ_2, Σ_2) du système (λ_1, Σ_1) construit par les lemmes de complétude de **Project**, **Imitate** et **Coalesce** (lemmes 4.34, 4.35 et 4.36), en posant $\Sigma_2 = \Phi_2;\ \Psi_2;\ \mathcal{C}_2;\ \mathcal{N}_2;\ \sigma_2$,

- d'après ces mêmes lemmes, la propriété 4.4.3 se propage à (λ_2, Σ_2) de sorte que pour tout $t \in \mathrm{st}_{\neq}(t_0)$,

$$\mathrm{R}^*(\lambda_2, \Phi\sigma_2, \Psi\sigma_2, t) = \emptyset$$

- $\Phi_2 = \Phi\sigma_2$ et $\Psi\sigma_2$ sont totalement stratifiés d'après le lemme 4.31 ;
- enfin, d'après le corollaire 4.25, justifié par la propriété (a) énoncée plus haut, la règle **Coalesce** (en particulier) ne s'applique qu'à des termes $t \in \mathrm{nps}(\Sigma_2)$ tels que $t\lambda_2 \in \mathrm{st}_{\neq}(t_0)$.

Ainsi, les conditions sémantique du lemme de complétude de **Coalesce** (*i.e.* portant sur la solution courante λ_2) sont toujours vérifiées lorsque cette règle s'applique syntaxiquement à un tel Σ_2. De plus, par le lemme 4.37, la règle **Fail** ne s'applique jamais. Par conséquent, l'une des deux règles **Project** et **Imitate** s'applique toujours si **Coalesce** ne s'applique pas et que le système n'est pas pré-résolu.

Finalement, sachant que les règles prioritaires terminent (section 4.1), on utilise successivement le lemme 4.36 (**Coalesce**), les lemmes 4.34 (**Project**) et 4.35 (**Imitate**), et enfin le lemme 4.43 (**Discard**) pour obtenir l'existence du couple (λ', Σ') voulu.

Dans le cas contraire, on a $\mathrm{R}(\lambda, \Sigma, t) = \emptyset$ pour tout terme t clos $\mathcal{R}$-réduit.

2. Supposons $\mathrm{t}(\lambda, \Sigma) = \emptyset$, *i.e.* tous les termes du premier ordre du système sont en forme normale. Comme Σ est pré-résolu et satisfiable, mais non résolu, il existe une équation $t_1 =^?_E t_2$ dans $\mathcal{C}$.

Sachant que $t_1\lambda$ et $t_2\lambda$ sont $\mathcal{R}$-réduits, le lemme 4.40 s'applique pour donner λ_1 et Σ_1 tels que

- $\Sigma \Longrightarrow_{\mathbf{Constrain}} \Sigma_1$,
- $\theta, \lambda_1 \models \Sigma_1$,
- $(\lambda, \Sigma) >_c (\lambda_1, \Sigma_1)$, et
- pour tout t $\mathcal{R}$-réduit, $\mathrm{R}(\lambda_1, \Sigma_1, t) \subseteq \mathrm{R}(\lambda, \Sigma, t) = \emptyset$.

Sachant que Σ_1 est totalement stratifié (lemme 4.31), le dernier point permet d'appliquer les lemmes de complétude des règles prioritaires comme ci-dessus pour obtenir les λ' et Σ' voulus.

3. Sinon, il existe un terme $t \in \mathrm{st}(\Sigma)$ tel que $t\lambda$ n'est pas $\mathcal{R}$-réduit. Considérons un tel terme t_0 pour lequel $t_0\lambda$ est minimal pour l'ordre sous-terme.

(a) Si $t_0 = x_0$ est une variable n'apparaissant pas dans $\mathcal{C}_{\triangleright}$ (ni dans Φ par l'invariant de stratification — proposition 4.19), alors le lemme 4.38 (**Normalize-free-variables**) s'applique pour donner les λ' et $\Sigma' = \Sigma$ recherchés.

(b) Si $t_0 = x_0$ est une variable apparaissant dans une contrainte $(X_0 \triangleright^? x_0) \in \mathcal{C}$ (c'est le cas si $x_0 \in \mathrm{var}(\Phi)$ par l'invariant de stratification (proposition 4.19)),

alors il existe un contexte public C_0, des règles $M_1 \rhd t_1, \ldots, M_m \rhd t_m$ telles que

$$\begin{aligned} X_0\lambda &= C_0[M_1, \ldots, M_m]\lambda \\ x_0\lambda &= C_0[t_1, \ldots, t_m]\lambda \end{aligned}$$

Comme $x_0\lambda$ est minimal, les $t_1\lambda, \ldots, t_m\lambda$ sont $\mathcal{R}$-réduits. Il existe donc une position p de profondeur maximale et interne à C_0 telle que $C_0[t_1, \ldots, t_m]|_p\lambda$ est réductible. Le lemme 4.41 s'applique donc pour donner λ_1 et Σ_1 tels que
- $\Sigma \Longrightarrow_{\mathbf{Context-1}} \Sigma_1$ ou $\Sigma \Longrightarrow_{\mathbf{Context-2}} \Sigma_1$,
- $\theta, \lambda_1 \models \Sigma_1$,
- $(\lambda, \Sigma) >_c (\lambda_1, \Sigma_1)$, et
- pour tout t_0 clos $\mathcal{R}$-réduit, $\mathrm{R}^*(\lambda_1, \Phi_1\mu, \Psi, t_0) = \emptyset$ où

$$\Phi_1 \triangleq \{(M \rhd t) \in \Phi \mid \mathrm{maxpar}(M) \leq \mathrm{ar}(X_0)\}$$

et les ensembles $\Phi_1\mu$ et Ψ sont totalement stratifiés et contenu dans Σ_1. Comme précédemment, le lemme 4.26 permet d'assurer dans ces conditions que les lemmes de complétude des règles prioritaires s'appliquent pour donner λ' et Σ'.

(c) Finalement, si t_0 est un terme non variable apparaissant dans une règle d'environnement de Φ ou dans une contrainte d'égalité de $\mathcal{C}$, alors comme $t_0\lambda$ est minimal et non variable, le lemme 4.39 s'applique pour donner λ_1 et Σ_1 tels que
- $\Sigma \Longrightarrow_{\mathbf{Narrowing}} \Sigma_1$,
- $\theta, \lambda_1 \models \Sigma_1$,
- $(\lambda, \Sigma) >_c (\lambda_1, \Sigma_1)$, et
- pour tout t_0 clos $\mathcal{R}$-réduit, $\mathrm{R}^*(\lambda_1, \Phi_1\mu, \Psi, t_0) \subseteq \mathrm{R}^*(\lambda, \Phi, \Psi, t_0) = \emptyset$ où

$$\begin{aligned} \Phi_0 &= \{(M \rhd t) \in \Phi \mid t_0 \in \mathrm{st}(t)\} \\ k_1 &= \min\{\mathrm{maxpar}(M) \mid (M \rhd t) \in \Phi_0\} \in [0, \infty] \\ \Phi_1 &= \{(M \rhd t) \in \Phi - \Phi_0 \mid \mathrm{maxpar}(M) \leq k_1\}\} \end{aligned}$$

et les ensembles $\Phi_1\mu$ et Ψ sont totalement stratifiés et contenu dans Σ_1. Comme plus haut, le lemme 4.27 garantit dans ces conditions que les lemmes de complétude des règles prioritaires s'appliquent pour donner λ' et Σ'. □

Lemme 4.45. *Soit Σ_0 un système initial, θ une solution de Σ_0, et $\Sigma_0 \Longrightarrow^* \Sigma$ une dérivation respectant les priorités entre règles, et telle que Σ est résolu et $\theta \models \Sigma$.*

Alors, il existe une dérivation standard $\Sigma_0 \Longrightarrow^* \Sigma'$ de longueur inférieure (ou égale), telle que Σ' est résolu et $\theta \models \Sigma'$.

Démonstration. On procède par récurrence sur la longueur de la dérivation $\Sigma_0 \Longrightarrow^* \Sigma$. Supposons que cette dérivation ne respecte pas la condition d'arrêt. Alors elle s'écrit $\Sigma_0 \Longrightarrow^* \Sigma_1 \Longrightarrow^+ \Sigma_2 \Longrightarrow^* \Sigma$ avec $\Sigma_1 \geq \Sigma_2$.

Par une récurrence évidente, la proposition 4.9 implique l'existence d'une dérivation $\Sigma_1 \Longrightarrow^* \Sigma'$ de même longueur que $\Sigma_2 \Longrightarrow^* \Sigma$ et telle que $\Sigma' \geq \Sigma$. En particulier, toujours d'après la proposition 4.9, $\Sigma' \geq \Sigma$ et les hypothèses du lemme impliquent que Σ' est résolu et $\theta \models \Sigma'$.

On conclut en appliquant l'hypothèse de récurrence à la dérivation $\Sigma_0 \Longrightarrow^* \Sigma_1 \Longrightarrow^* \Sigma'$ (strictement plus courte que la dérivation initiale $\Sigma_0 \Longrightarrow^* \Sigma$). □

Fin de la preuve de complétude. Nous pouvons maintenant conclure la preuve de la complétude des dérivations standard (proposition 4.30). En effet, étant donné un système initial (et donc pré-résolu) Σ_0, la seule règle prioritaire pouvant s'appliquer est **Discard**, et de même jusqu'à ce que le système soit saturé par les règles prioritaires. On établit donc la complétude des dérivations respectant les priorités entre règles en appliquant le lemme de complétude de **Discard** (lemme 4.43), puis le lemme de progression (lemme 4.44). À partir de là, la complétude des dérivations standard, c'est-à-dire respectant en plus la condition d'arrêt, découle du lemme 4.45.

4.5 Terminaison

Nous établissons la terminaison de la procédure dans le cas d'un système de réécriture $\mathcal{R}$ sous-terme-convergent et fini.

4.5.1 Notion de taille des systèmes de contraintes

Soit $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ un système de contraintes. La représentation concrète que nous considérons pour un tel système au cours de l'algorithme peut être décrite (succinctement) de la manière suivante.

- Les sous-termes de Φ sont organisés sous forme de deux forêts distinctes avec partage total. La première, Σ^1, correspond aux termes du premier ordre notés t dans la définition 4.1, tandis que la seconde, Σ^2, correspond aux termes du second ordre notés M. Les tailles de ces forêts sont notées respectivement

$$|\Sigma^1|_{\mathrm{st}} \triangleq |\Sigma|^1_{\mathrm{st}} \triangleq \mathrm{card}\{t \in \mathrm{st}(\Sigma)\}$$
$$|\Sigma^2|_{\mathrm{st}} \triangleq |\Sigma|^2_{\mathrm{st}} \triangleq \mathrm{card}\{M \in \mathrm{st}(\Sigma)\}$$

  où, dans la seconde équation, on étend la notion de sous-termes à chaque équation de Ψ par

$$\mathrm{st}(\forall \beta.M \bowtie N) = \mathrm{st}(M, N)$$

  pour un certain choix de variables fraîches β.
- Chaque règle d'environnement $(M \rhd t) \in \Phi$ est donnée par un couple de sommets respectivement dans Σ^2 et Σ^1. On représente de la même manière les équations $\forall \beta.M \bowtie N$ de Ψ, les contraintes $t_1 =^?_E t_2$, $t_1 \neq_E t_2$, $X \rhd^? t$ dans $\mathcal{C}$, tandis chaque terme de $\mathcal{N}$ est donné par un sommet de Σ^1.
- Enfin, pour tout $V \in \mathrm{var}(\Sigma)\,(\supseteq \mathrm{supp}(\sigma))$, on représente le couple $(V, V\sigma)$ par un couple de sommets dans Σ^1 ou dans Σ^2, selon la nature de V.

Compte tenu de cette représentation, nous utilisons la notion suivante de taille pour les système de contraintes :

$$|\Sigma| \triangleq |\Sigma|^1_{\mathrm{st}} + |\Sigma|^2_{\mathrm{st}} + 2\,\mathrm{card}(\Phi) + 2\,\mathrm{card}(\Psi) + 2\,\mathrm{card}(\mathcal{C}) + \mathrm{card}(\mathcal{N}) + \mathrm{card}(\mathrm{var}(\Sigma))$$

De la même manière, nous supposons que les règles de $\mathcal{R}$ sont préalablement décrites sous forme de graphe avec partage total et nous définissons :

$$|\mathcal{R}|_{\mathrm{st}} \triangleq \mathrm{card}\{t \in \mathrm{st}(\mathcal{R})\}$$

Notons que pour un système $\mathcal{R}$ fixé, il existe un nombre fini de décompositions $l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$ pour $l \to r \in \mathcal{R}$.

En effet, ce nombre est borné (grossièrement) par la quantité

$$\sum_{l \to r \in \mathcal{R}} 2^{|l|}$$

où $|l| = \mathrm{card}(\mathrm{pos}(l))$ est la taille du terme l (sans partage).

En ce qui concerne l'analyse précise de la complexité (en vue de la section 4.6), nous supposerons que les unificateurs plus généraux (mgu) sont calculés à l'aide d'un algorithme polynomial, compatible avec la représentation des termes avec partage total. (En particulier, l'application d'un tel mgu n'augmente pas la taille DAG du système.) Un tel algorithme est décrit par exemple par Corbin et Bidoit [CB83].

4.5.2 Mesure de terminaison syntaxique

Nous décrivons maintenant une mesure permettant de borner l'application des règles principales autres que **Discard**.

Soit $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ un système de contraintes. Rappelons que la notation $\Phi^{\mathcal{N}}$ désigne l'ensemble

$$\Phi^{\mathcal{N}} \triangleq \{(M \rhd t) \in \Phi \mid t \in \mathrm{st}(\mathcal{N}) \cup \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}}\}$$

Nous notons également $\mathcal{R}^{\emptyset}$ l'ensemble des membres droits de règles $l \to r$ dans $\mathcal{R}$ tels que r est clos $\mathcal{R}$-réduit :

$$\mathcal{R}^{\emptyset} \triangleq \{\, r \mid (l \to r) \in \mathcal{R},\, r \in \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}}\}$$

Le système $\mathcal{R}$ étant sous-terme-convergent, on a par définition : pour toute règle $(l \to r) \in \mathcal{R}$, $r \in \mathrm{st}_{\neq}(l) \cup \mathcal{R}^{\emptyset}$.

Termes surréductibles

Un *terme surréductible* de Σ est un sous-terme $t \in \mathrm{st}^1(\Phi; \mathcal{C}) - \mathrm{st}(\mathcal{N}) - \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}}$. On note $\mathrm{Narr}(\Sigma)$ l'ensemble des termes surréductibles de Σ, et l'on pose

$$\mathrm{narr}(\Sigma) \triangleq \mathrm{card}(\mathrm{Narr}(\Sigma))$$

En particulier, pour tout Σ, $\mathrm{narr}(\Sigma) \leq |\Sigma|^1_{\mathrm{st}}$.

Contraintes d'égalité

Le cardinal de $\mathcal{C}_{=} \triangleq \{t =^?_E t' \in \mathcal{C}\}$ est notée $\mathrm{c}(\Sigma)$ comme précédemment.

Notons que $\mathrm{c}(\Sigma) \leq (|\Sigma|^1_{\mathrm{st}})^2$.

Visibilité des secrets

Nous définissons ici un indicateur permettant de mesurer la progression de l'algorithme en terme d'apprentissage de nouvelles règles d'environnement $M \rhd t$.

Tout d'abord, soit $\mathrm{Secr}(\Sigma)$ l'ensemble des *secrets* de Σ défini de la manière suivante : $t \in \mathrm{Secr}(\Sigma)$ ssi $t \in \mathcal{R}^{\emptyset}$ ou s'il existe une règle $(M_0 \rhd t_0) \in \Phi^{\mathcal{N}}$ vérifiant

(i) $t \in \mathrm{st}_{\neq}(t_0)$,

(ii) pour tout $(X \rhd^{?} s) \in \mathcal{C}$, $X \preceq M_0 \Rightarrow t \notin \mathrm{st}_{\neq}(s)$,

(iii) pour tout N, $N \preceq M_0 \Rightarrow N \not\rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t$

Par secret, on entend ici un terme « intéressant à deviner » pour l'attaquant.

Un terme t est *visible au niveau k dans* Σ ($0 \leq k \leq k_0$) ss'il existe un terme N tel que

- $N \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t$,
- $\mathrm{maxar}(N) < k$ et
- $\mathrm{maxpar}(N) \leq k$.

Notons que cette notion s'entend dans la théorie équationnelle vide, et concerne seulement les règles d'environnements surréduites.

Le *niveau* d'un terme t dans Σ est défini par

$$\ell(\Sigma, t) \stackrel{\Delta}{=} \min \{k \mid t \text{ est visible au niveau } k \text{ dans } \Sigma\} \cup \{k_0 + 1\}.$$

On a donc $\ell(\Sigma, t) = k_0 + 1$ ssi t n'est visible à aucun niveau dans Σ.

Enfin, le *niveau global* de Σ est défini par

$$\ell(\Sigma) \stackrel{\Delta}{=} \sum_{t \in \mathrm{Secr}(\Sigma)} \ell(\Sigma, t)$$

En particulier, pour tout Σ, $\ell(\Sigma) \leq (k_0 + 1)\,(|\Sigma|^1_{\mathrm{st}} + |\mathcal{R}^{\emptyset}|_{\mathrm{st}})$.

Termes connus substituables par Context-$\{1, 2\}$

On note également $\mathrm{Nmatch}(\Sigma)$ l'ensemble des couples (t, l') tels que t est un membre droit de règle d'environnement $(M \rhd t) \in \Phi^{\mathcal{N}}$, $l' \in \mathrm{st}_{\neq}(l)$ est sous-terme strict de membre gauche d'une règle $(l \to r) \in \mathcal{R}$, et tel que t n'est pas une instance de l' :

$$\begin{aligned} \mathrm{Nmatch}(\Sigma) \quad \stackrel{\Delta}{=} \quad \{\, (t, l') \mid (M \rhd t) \in \Phi^{\mathcal{N}}, \quad (l \to r) \in \mathcal{R}, \\ l' \in \mathrm{st}_{\neq}(l), \quad \forall \mu,\, l'\mu \neq t \,\} \end{aligned}$$

Le cardinal de cet ensemble est noté $\mathrm{nmatch}(\Sigma)$ et vérifie

$$\mathrm{nmatch}(\Sigma) \;\leq\; |\Sigma|^1_{\mathrm{st}}\, |\mathcal{R}|_{\mathrm{st}}.$$

Variables du premier ordre non résolues

L'ensemble des variables du premier ordre dans Φ et $\mathcal{C}$ est noté

$$\mathrm{Nres}(\Sigma) \stackrel{\Delta}{=} \mathrm{var}^1(\Phi; \mathcal{C})$$

et l'on pose

$$\mathrm{nres}(\Sigma) = \mathrm{card}(\mathrm{Nres}(\Sigma))$$

En particulier, pour tout Σ, $\mathrm{nres}(\Sigma) \leq |\Sigma|^1_{\mathrm{st}}$.

Notons que pour un système Σ stratifié (au sens de la proposition 4.19), on a en fait $\mathrm{Nres}(\Sigma) = \mathrm{var}^1(\mathcal{C})$.

Équations directement ajoutables

Une équation est *directement ajoutable* dans Σ ssi elle n'appartient pas à Ψ et elle est d'une des formes suivantes :

(a) $M \bowtie N$ où il existe t telle que $(M \rhd t) \in \Phi$ et $N \rhd_{\Phi,\mathcal{C}} t$; (**Discard**)

(b) $\forall Z_1, \ldots, Z_q.M \bowtie N$ où il existe une règle $(l \to r) \in \mathcal{R}$, une décomposition propre $l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$, et une substitution μ tels que
- $\mathrm{supp}(\mu) \subseteq \mathrm{var}(l) - \{z_1, \ldots, z_q\}$,
- $\mathrm{var}(\{l_1, \ldots, l_n, y_1, \ldots, y_p\}\mu) \subseteq \mathrm{var}(\Sigma)$,
- les variables $Z_1, \ldots, Z_q$, sont deux à deux distinctes, d'arité k_0 et de sortes respectives $\mathrm{sort}(Z_i) = \mathrm{sort}(z_i)$,
- M est de la forme $D[M_1, \ldots, M_n, N_1, \ldots, N_p, Z_1 \ldots Z_q]$ où pour tout $1 \leq i \leq n$, $M_i \rhd l_i\mu$ est dans Φ, et pour tout $1 \leq j \leq p$, on a $N_j \rhd_{\Phi,\mathcal{C}} y_j\mu$,
- N vérifie $N \rhd_{\Phi,\mathcal{C}\cup\{Z_k \rhd^? z_k\}_{1\leq k\leq q}} r\mu$; (**Context-2**)

(c) $\forall Z_1, \ldots, Z_q.M \bowtie N$ où il existe une règle $(l \to r) \in \mathcal{R}$, une décomposition propre $l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$, et une substitution μ tels que
- $\mathrm{supp}(\mu) \subseteq \mathrm{var}(l) - \{z_1, \ldots, z_q\}$,
- $\mathrm{var}(\{l_1, \ldots, l_n, y_1, \ldots, y_p\}\mu) \subseteq \mathrm{var}(\Sigma)$,
- les variables $Z_1, \ldots, Z_q$, sont deux à deux distinctes, d'arité k_0 et de sortes respectives $\mathrm{sort}(Z_i) = \mathrm{sort}(z_i)$,
- M est de la forme $D[M_1, \ldots, M_n, N_1, \ldots, N_p, Z_1 \ldots Z_q]$ où pour tout $1 \leq i \leq n$, $M_i \rhd l_i\mu$ est dans Φ, et pour tout $1 \leq j \leq p$, on a $N_j \rhd_{\Phi,\mathcal{C}} y_j\mu$,
- $r \in \mathrm{st}_{\neq}(l_1, \ldots, l_n) \cup \mathcal{R}^{\emptyset}$ et $N = M\{Z_k \mapsto \mathsf{c}_{\mathrm{sort}(Z_k)}\}_{1\leq k\leq q}$. (**Context-1**)

L'ensemble des équations directement ajoutables dans Σ est noté $\mathrm{Qadd}(\Sigma)$.

Intuitivement, $\mathrm{Qadd}(\Sigma)$ contient les équations ajoutables par les règles de transformation sans substituer les variables du premier ordre du problème (et plus précisément les variables dans $\mathrm{var}^1(\Phi)$).

Lemme 4.46. *Pour tout système Σ, l'ensemble $\mathrm{Qadd}(\Sigma)$ est fini.*

Démonstration. En effet, dans le cas (a), pour un terme t donné, le cardinal de l'ensemble $\{N \mid N \rhd_{\Phi,\mathcal{C}} t\}$ est borné (grossièrement) par

$$(\mathrm{card}(\Phi) + \mathrm{card}(\mathcal{C}) + 1)^{|t|}$$

où $|t| \triangleq \mathrm{card}(\mathrm{pos}(t))$.

Concernant les points (b) et (c), nous avons vu que le nombre de décompositions de règles de $\mathcal{R}$ est fini. Or, étant donnée une décomposition $l = D[l_1, \ldots, l_n, y_1, \ldots, y_p, z_1, \ldots, z_q]$, le nombre de substitutions partielles $\mu|_{\mathrm{var}(l_1,\ldots,l_n)}$ telles que pour tout $1 \leq i \leq n$, il existe M_i avec $M_i \rhd l_i\mu$ est dans Φ est borné par $\mathrm{card}(\Phi)^n$.

Pour une telle substitution, du fait que $y_1, \ldots, y_p \in \mathrm{var}(l_1, \ldots, l_n)$, les valeurs $y_j\mu$ sont fixées et par conséquent pour chaque j, le nombre de N_j tels que $N_j \rhd_{\Phi,\mathcal{C}} y_j\mu$ est borné par $(\mathrm{card}(\Phi) + \mathrm{card}(\mathcal{C}) + 1)^{|y_j\mu|}$.

Enfin, dans le cas (b), le nombre de N tel que $N \rhd_{\Phi,\mathcal{C}\cup\{Z_k \rhd^? z_k\}_{1\leq k\leq q}} r\mu$ est borné de la même manière. □

Dans la suite, on note $\mathrm{qadd}(\Sigma) \triangleq \mathrm{card}(\mathrm{Qadd}(\Sigma))$.

Arité des variables du second-ordre

Enfin, la somme des arités des variables du second-ordre non résolues de Σ est notée

$$\mathrm{sar}(\Sigma) \stackrel{\triangle}{=} \sum_{X \in \mathrm{var}^*(\Sigma)} \mathrm{ar}(X)$$

où l'on a défini précédemment (sous-section 4.1.3) : $\mathrm{var}^*(\Sigma) \stackrel{\triangle}{=} \mathrm{var}(\Phi;\, \Psi;\, \mathcal{C};\, \mathcal{N};\, \mathcal{Y}\sigma)$.

Mesure obtenue

Définition 4.14. La mesure de terminaison sur les configurations Σ est définie par

$$\mathrm{m}_t(\Sigma) = (\mathrm{narr}(\Sigma),\, \mathrm{c}(\Sigma),\, \ell(\Sigma),\, \mathrm{nmatch}(\Sigma), \mathrm{nres}(\Sigma),\, \mathrm{qadd}(\Sigma),\, \mathrm{sar}(\Sigma))$$

On note $\Sigma \geq_t \Sigma'$ lorsque $\mathrm{m}_t(\Sigma)$ est plus grand que $\mathrm{m}_t(\Sigma')$ pour l'ordre lexicographique sur les 7-uplets d'entiers. L'ordre strict associé à $\geq_t$ est noté $>_t$, tandis que la relation d'équivalence correspondante est notée $\equiv_t$: $\Sigma \equiv_t \Sigma'$ ssi $\mathrm{m}_t(\Sigma) = \mathrm{m}_t(\Sigma')$.

Propriétés remarquables

Nous terminons cette section par plusieurs lemmes techniques concernant les éléments de mesure $\mathrm{narr}(\Sigma)$, $\mathrm{nres}(\Sigma)$ et $\ell(\Sigma)$.

Lemme 4.47. *Soit $\Sigma = \Phi;\, \Psi;\, \mathcal{C};\, \mathcal{N};\, \sigma$ un système de contraintes, μ une substitution (bien formée) de support inclus dans $\mathcal{X}^1$, et $\mathcal{N}_1$ un ensemble de termes du premier ordre tel que $\mathcal{N}\mu \subseteq \mathcal{N}_1$ et $\mathrm{supp}(\mu)\mu \subseteq \mathrm{st}(\mathcal{N}_1)$. Soit $\Sigma_1 = \Phi\mu;\, \Psi\mu;\, \mathcal{C}\mu;\, \mathcal{N}_1;\, \sigma\mu$. On a*

$$\begin{aligned} (\Phi^{\mathcal{N}})\mu &\subseteq (\Phi\mu)^{\mathcal{N}_1} \\ \mathrm{Narr}(\Sigma_1) &\subseteq \mathrm{Narr}(\Sigma)\mu \end{aligned}$$

De plus, si $\mathrm{narr}(\Sigma_1) = \mathrm{narr}(\Sigma)$, alors

$$(\Phi^{\mathcal{N}})\mu = (\Phi\mu)^{\mathcal{N}_1}$$

Démonstration. $(\Phi^{\mathcal{N}})\mu \subseteq (\Phi\mu)^{\mathcal{N}_1}$ est clair du fait que $\mathcal{N}_1 \supseteq \mathcal{N}\mu$.

Montrons que $\mathrm{Narr}(\Sigma_1) \subseteq \mathrm{Narr}(\Sigma)\mu$. En effet, soit $t' \in \mathrm{Narr}(\Sigma_1) = \mathrm{st}^1(\Phi\mu, \mathcal{C}\mu) - \mathrm{st}(\mathcal{N}_1) - \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}}$. Comme $\mathrm{supp}(\mu)\mu \subseteq \mathrm{st}(\mathcal{N}_1)$, il existe $t \in \mathrm{st}^1(\Phi, \mathcal{C})$ tel que $t' = t\mu$. Du fait que $t' = t\mu \notin \mathrm{st}(\mathcal{N}\mu) \cup \mathcal{F}[\emptyset]\downarrow_{\mathcal{R}}$, on a bien $t \notin \mathrm{st}(\mathcal{N}) \cup \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}}$.

Si $\mathrm{narr}(\Sigma_1) = \mathrm{narr}(\Sigma)$, alors d'après ce qui précède $\mathrm{Narr}(\Sigma_1) = \mathrm{Narr}(\Sigma)\mu$. Autrement dit, pour $t \in \mathrm{st}^1(\Phi; \mathcal{C})$, on a l'implication

$$t\mu \in \mathrm{st}(\mathcal{N}_1) \cup \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}} \Rightarrow\ t \in \mathrm{st}(\mathcal{N}) \cup \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}}$$

En particulier, $(\Phi\mu)^{\mathcal{N}_1} \subseteq (\Phi^{\mathcal{N}})\mu$. □

Lemme 4.48. *Soit $\Sigma = \Phi;\, \Psi;\, \mathcal{C};\, \mathcal{N};\, \sigma$ un système de contraintes, et μ une substitution idempotente bien formée telle que $\mathrm{var}(\mu) \stackrel{\triangle}{=} \mathrm{supp}(\mu) \uplus \mathrm{var}(\mathrm{supp}(\mu)\mu) \subseteq \mathrm{Nres}(\Sigma) \stackrel{\triangle}{=} \mathrm{var}^1(\Phi;\, \mathcal{C})$. On a*

$$\mathrm{Nres}(\Sigma\mu) \subseteq \mathrm{Nres}(\Sigma)\mu$$

De plus, si $\mathrm{nres}(\Sigma\mu) = \mathrm{nres}(\Sigma)$, alors μ est l'identité.

Démonstration. Soit $x' \in \mathrm{Nres}(\Sigma\mu)$. Il existe $x \in \mathrm{Nres}(\Sigma)$ tel que $x' \in \mathrm{var}(x\mu)$. Si $x \notin \mathrm{supp}(\mu)$, alors $x' = x \in \mathrm{Nres}(\Sigma) - \mathrm{supp}(\mu)$. Dans le cas contraire, $x' \in \mathrm{var}(x\mu) \subseteq \mathrm{var}(\mathrm{supp}(\mu)\mu) \subseteq \mathrm{Nres}(\Sigma) - \mathrm{supp}(\mu)$.

Si $\mathrm{nres}(\Sigma\mu) = \mathrm{nres}(\Sigma)$, alors $\mathrm{supp}(\mu) \cap \mathrm{Nres}(\Sigma) = \emptyset$. Or par hypothèse, $\mathrm{supp}(\mu) \subseteq \mathrm{Nres}(\Sigma)$, donc $\mathrm{supp}(\mu) = \emptyset$. □

Lemme 4.49. *Soit* $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ *et* $\Sigma' = \Phi'; \Psi'; \mathcal{C}'; \mathcal{N}'; \sigma'$ *deux systèmes de contraintes tels que*

(1) Σ *est stratifié ;*

(2) $\mathrm{var}(\Sigma) \subseteq \mathrm{var}(\Sigma')$ *et* $\sigma\sigma' = \sigma'$*,*

(3) l'une des deux conditions suivante est vraie :

- *(3a) pour tout* M *et* t *tels que* $M \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t$*, on a*
$$M\sigma' \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t\sigma'$$
- *(3b)* $\mathcal{C}' = \mathcal{C}$*,* $\sigma' = \sigma$*, et pour tout* M *et* t *tels que* $M \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t$*, il existe* N *tel que*
$$M \succeq N \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t$$

(4) $\Phi'^{\mathcal{N}'} \subseteq (\Phi^{\mathcal{N}})\sigma'$*.*

Alors, on a $\mathrm{Secr}(\Sigma') \subseteq \mathrm{Secr}(\Sigma)\sigma$*.*

Notons que les conditions (1), (2) et (3) découlent en particulier de la proposition 4.19 et du lemme 4.21 lorsque Σ est un système accessible, dans les cas suivants :

- $\Sigma \Longrightarrow^* \Sigma'$ par une dérivation n'utilisant pas la règle **Discard** (cas (3a)) ;
- $\Sigma \Longrightarrow^* \Sigma'$ par une dérivation utilisant seulement la règle **Discard** (cas (3b)) ;
- $\Sigma' = \Sigma\mu$ pour une substitution (bien formée) μ de support inclus dans $\mathcal{X}^1$ (cas (3a)).

Démonstration. Soit $\prec$ le pré-ordre bien fondé défini par : $M \prec N$ ssi $M \in \mathrm{st}_{\neq}(N)$ ou $\max(\mathrm{maxpar}(M), \mathrm{maxar}(M)) < \mathrm{maxpar}(N)$.

Notons que $\prec$ est bien transitif et inclus dans $\preceq$.

Nous montrons tout d'abord la propriété suivante par induction sur $M_0\sigma'$ muni de $\prec$:

> (P) pour tous M_0 et t_0 tels que $M_0 \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t_0$, et pour tout $t \in \mathrm{st}(t_0\sigma')$, l'une des conditions suivantes est vérifiée :
>
> (a) il existe $(M_1 \rhd t_1) \in \Phi^{\mathcal{N}}$ tel que $t \in \mathrm{st}_{\neq}(t_1)\sigma'$ et $M_1\sigma' \preceq M_0\sigma'$.
>
> (b) il existe $(X \rhd^? s) \in \mathcal{C}'$ tel que $t \in \mathrm{st}_{\neq}(s)$ et $X \preceq M_0\sigma'$;
>
> (c) il existe N tel que $N \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t$ et $N \preceq M_0\sigma'$.

De plus, dans le cas (3a), nous montrons la variante suivante :

> (P') pour tous M_0 et t_0 tels que $M_0 \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t_0$, et pour tout $t \in \mathrm{st}(t_0\sigma')$, l'une des conditions suivantes est vérifiée :
>
> (a) il existe $(M_1 \rhd t_1) \in \Phi^{\mathcal{N}}$ tel que $t \in \mathrm{st}_{\neq}(t_1)\sigma'$ et $M_1\sigma'$ $(\prec \cup =)$ $\mathrm{st}(M_0\sigma')$;
>
> (b) il existe $(X \rhd^? s) \in \mathcal{C}'$ tel que $t \in \mathrm{st}_{\neq}(s)$ et $X \preceq M_0\sigma'$;
>
> (c) il existe N tel que $N \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t$ et $N \preceq M_0\sigma'$.

Supposons $M_0 \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t_0$. Par hypothèse, on a $M_0\sigma' \succeq N_0 \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t\sigma'$ pour un certain N_0 vérifiant $N_0 \preceq M_0\sigma'$, et $N_0 \neq M_0\sigma' \Rightarrow \sigma = \sigma'$.

Comme $\Phi'^{\mathcal{N}'} \subseteq (\Phi^{\mathcal{N}})\sigma'$, il existe donc un contexte public C, des règles $M_1 \rhd t_1$, $\ldots$, $M_m \rhd t_m$ dans $\Phi^{\mathcal{N}}$, et des contraintes $X_1 \rhd^? s_1$, $\ldots$, $X_n \rhd^? s_n$ dans $\mathcal{C}'$ telles que

$$\begin{aligned} N_0 &= C[M_1\sigma', \ldots, M_m\sigma', X_1, \ldots, X_n] \\ t_0\sigma' &= C[t_1\sigma', \ldots, t_m\sigma', s_1, \ldots, s_n] \end{aligned}$$

Comme $t \in \mathrm{st}(t_0\sigma')$, on en déduit que l'un des trois cas suivants se présente :
- Le point (b) de (P) et (P') est vérifié pour un certain $X_i \rhd^? s_i$ dans $\mathcal{C}'$ vérifiant $t \in \mathrm{st}_{\neq}(s_i)$ et $X_i \preceq N_0 \preceq M_0\sigma'$.
- Le point (c) de (P) et (P') est vérifié pour un certain terme $N = C'[M_1\sigma', \ldots, M_m\sigma', X_1, \ldots, X_n] \preceq N_0 \preceq M_0\sigma'$ où $C' \in \mathrm{st}(C)$.
- Il existe une règle $M_i \rhd t_i$ dans $\Phi^{\mathcal{N}}$ telle que $t \in \mathrm{st}_{\neq}(t_i\sigma')$ et $M_i\sigma' \in \mathrm{st}(N_0)$.

Dans le dernier cas, notons que $t \in \mathrm{st}_{\neq}(t_i\sigma') \subseteq \mathrm{st}_{\neq}(t_i)\sigma' \cup \mathrm{st}(\mathrm{var}(t_i)\sigma')$.

Si $t \in \mathrm{st}_{\neq}(t_i)\sigma'$, le point (a) de (P) est vérifié car $M_i\sigma' \in \mathrm{st}(N_0)$ et $N_0 \preceq M_0\sigma'$ (et (P') également si $N_0 = M_0\sigma'$).

Dans le cas contraire, $t \in \mathrm{st}(\mathrm{var}(t_i)\sigma') - \mathrm{st}_{\neq}(t_i\sigma')$. Nécessairement $\sigma' \neq \sigma$ donc $N_0 = M_0\sigma'$. Montrons le point (a) de (P').

Soit $x \in \mathrm{var}(t_i)$ telle que $t \in \mathrm{st}(x\sigma')$. Comme Σ est stratifié, M_i n'est pas une variable, et il existe une contrainte $X_0 \rhd^? s_0$ dans $\mathcal{C}$ telle que $x \in \mathrm{var}(s_0)$ et $X_0 \preceq M_i$.

On a donc $X_0 \in \mathrm{st}_{\neq}(M_i)$ ou $\mathrm{ar}(X_0) < \mathrm{maxpar}(M_i)$. Dans les deux cas, sachant que σ est bien formée et que $M_i\sigma' \in \mathrm{st}(M_0\sigma')$, on obtient $X_0\sigma' \in \mathrm{st}_{\neq}(M_0\sigma')$ ou $\max(\mathrm{maxpar}(X_0\sigma'), \mathrm{maxar}(X_0\sigma')) < \mathrm{maxpar}(M_0\sigma')$; d'où $X_0\sigma' \prec M_0\sigma'$.

Or, d'une part $t \in \mathrm{st}(\mathrm{var}(s)\sigma') \subseteq \mathrm{st}(s\sigma')$ et, d'autre part, $X \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} s$.

Par conséquent, l'hypothèse d'induction de (P') s'applique à $X_0\sigma'$ pour donner trois cas possibles :

(a) il existe $(M' \rhd t') \in \Phi^{\mathcal{N}}$ tel que $t \in \mathrm{st}_{\neq}(t')\sigma'$ et $M'\sigma'$ $(\prec \cup =)$ $\mathrm{st}(M'\sigma')$; d'où on déduit $M'\sigma'(\prec \cup =)M_0\sigma'$.

(b) il existe $(X \rhd^? s) \in \mathcal{C}'$ tel que $t \in \mathrm{st}_{\neq}(s)$ et $X \preceq X_0\sigma' \preceq M_0\sigma'$;

(c) il existe N tel que $N \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t$ et $N \preceq X_0\sigma' \preceq M_0\sigma'$.

Les propriétés (P) et (P') sont donc démontrées. Montrons que $\mathrm{Secr}(\Sigma') \subseteq \mathrm{Secr}(\Sigma)\sigma'$.

Soit $t' \in \mathrm{Secr}(\Sigma')$. Si $t' \in \mathcal{R}^{\emptyset}$ alors $t' \in \mathrm{Secr}(\Sigma)\sigma'$. Sinon, comme $\Phi'^{\mathcal{N}'} \subseteq \Phi^{\mathcal{N}}\sigma'$, il existe une règle $(M_0 \rhd t_0) \in \Phi^{\mathcal{N}}$ vérifiant les conditions suivantes :

(i) $t' \in \mathrm{st}_{\neq}(t_0\sigma')$,

(ii) pour tout $(X \rhd^? s) \in \mathcal{C}'$, $X \preceq M_0\sigma' \Rightarrow t' \notin \mathrm{st}_{\neq}(s)$,

(iii) pour tout N, $N \preceq M_0\sigma' \Rightarrow N \not\rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t'$

Comme en particulier $M_0 \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t_0$ et $t' \in \mathrm{st}(t_0\sigma')$, la propriété (P) nous donne :

(a) il existe $(M_1 \rhd t_1) \in \Phi^{\mathcal{N}}$ tel que $t' \in \mathrm{st}_{\neq}(t_1)\sigma'$ et $M_1\sigma' \preceq M_0\sigma'$; ou bien

(b) il existe $(X \rhd^? s) \in \mathcal{C}'$ tel que $t' \in \mathrm{st}_{\neq}(s)$ et $X \preceq M_0\sigma'$; ou bien

(c) il existe N tel que $N \rhd_{\Phi'^{\mathcal{N}'},\mathcal{C}'} t'$ et $N \preceq M_0\sigma'$.

(b) et (c) étant en contradiction avec (ii) et (iii), le point (a) est vérifié. Soit donc $(M_1 \rhd t_1) \in \Phi^{\mathcal{N}}$ tel que $M_1\sigma'$ est minimal pour le pré-ordre $\prec$, et $t \in \mathrm{st}_{\neq}(t_1)$ tels que $t' = t\sigma'$ et $M_1\sigma' \preceq M_0\sigma'$.

Nous vérifions les conditions restantes pour avoir $t \in \mathrm{Secr}(\Sigma)$.

Soit N tel que $N \preceq M_1$. Supposons $N \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t$. Alors par hypothèse on en déduit qu'il existe N' tel que

$$M_1\sigma' \succeq N\sigma' \succeq N' \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t\sigma'$$

ce qui contredit le point (iii).

Soit $(X \rhd^? s) \in \mathcal{C}$ tel que $X \preceq M_1$, c'est-à-dire $X \prec M_1$ car M_1 n'est pas une variable par stratification de Σ. Supposons $t \in \mathrm{st}_{\neq}(s)$. Dans le cas (3b), comme $\mathcal{C} = \mathcal{C}'$ et $M_1\sigma' = M_1$, ceci contredit le point (ii).

Dans le cas (3a), comme en particulier $X \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} s$ et $t' \in \mathrm{st}(s\sigma')$, la propriété (P') nous donne :

(a) il existe $(M_2 \rhd t_2) \in \Phi^{\mathcal{N}}$ tel que $t' \in \mathrm{st}_{\neq}(t_2)\sigma'$ et $M_2\sigma'$ $(\prec \cup =)$ $\mathrm{st}(X\sigma')$;

(b) il existe $(X' \rhd^? s') \in \mathcal{C}'$ tel que $t' \in \mathrm{st}_{\neq}(s')$ et $X' \preceq X\sigma' \preceq M_1\sigma'$; ou bien

(c) il existe N tel que $N \rhd_{\Phi'\mathcal{N}',\mathcal{C}'} t'$ et $N \preceq X\sigma' \preceq M_1\sigma'$.

Les cas (b) et (c) contredisent les points (ii) et (iii). Dans le cas (a), comme $X\sigma' \prec M_1\sigma'$, on obtient $M_2\sigma' \prec M_1\sigma'$, ce qui contredit la minimalité de $M_1\sigma'$.

Par conséquent, $\mathrm{Secr}(\Sigma') \subseteq \mathrm{Secr}(\Sigma)\sigma'$. □

Lemme 4.50. *Soit $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ et $\Sigma' = \Phi'; \Psi'; \mathcal{C}'; \mathcal{N}'; \sigma'$ deux systèmes de contraintes vérifiant $\mathrm{Secr}(\Sigma') \subseteq \mathrm{Secr}(\Sigma)\sigma'$ et tels que pour tous N, t vérifiant $N \rhd_{\Phi,\mathcal{C}} t$, il existe N' tel que $N\sigma' \succeq N' \rhd_{\Phi'\mathcal{N}',\mathcal{C}'} t\sigma'$.*

Alors on a $\ell(\Sigma') \leq \ell(\Sigma)$, et pour tout $t \in Secr(\Sigma)$, $\ell(\Sigma', t\sigma') \leq \ell(\Sigma, t)$.

De plus, si $\ell(\Sigma') = \ell(\Sigma)$, alors pour tout $t \in Secr(\Sigma)$, $\ell(\Sigma', t\sigma') = \ell(\Sigma, t)$.

Démonstration. Soit $t \in \mathrm{Secr}(\Sigma)$ et $k = \ell(\Sigma, t)$. On peut supposer $k \leq k_0$ (sinon $\ell(\Sigma', t\sigma') \leq k$ est clair). Il existe donc N tel que $N \rhd_{\Phi^{\mathcal{N}},\mathcal{C}} t$ et $\mathrm{maxpar}(N) \leq k$, $\mathrm{maxar}(N) < k$. Par hypothèse, ceci implique $N\sigma' \succeq N' \rhd_{\Phi'\mathcal{N}',\mathcal{C}'} t\sigma'$ pour un certain N'. Par bonne formation de σ', et par définition de $\preceq$, on en déduit

$$\mathrm{maxpar}(N') \leq \mathrm{maxpar}(N\sigma') \leq \mathrm{maxpar}(N) \leq k$$

et

$$\mathrm{maxar}(N') \leq \max(\mathrm{maxpar}(N\sigma') - 1, \mathrm{maxar}(N\sigma')) < k$$

soit enfin $\ell(\Sigma', t\sigma') \leq k = \ell(\Sigma, t)$.

Par conséquent, on a la suite d'inégalités :

$$\ell(\Sigma') = \sum_{t' \in \mathrm{Secr}(\Sigma')} \ell(\Sigma', t') \leq \sum_{t \in \mathrm{Secr}(\Sigma)} \ell(\Sigma', t\sigma') \leq \sum_{t \in \mathrm{Secr}(\Sigma)} \ell(\Sigma, t) = \ell(\Sigma)$$

Enfin, si $\ell(\Sigma') = \ell(\Sigma)$, alors les inégalités ci-dessus étant atteintes, on a en particulier pour tout $t \in \mathrm{Secr}(\Sigma)$, $\ell(\Sigma', t\sigma') = \ell(\Sigma, t)$. □

4.5.3 Règles prioritaires

Nous avons déjà montré que les règles prioritaires terminaient. Les lemmes suivants précisent le comportement de ces règles vis-à-vis de la mesure et de la taille des systèmes de contraintes.

Lemme 4.51. *Soit* $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ *un système de contraintes. Pour toute dérivation* $\Sigma \Longrightarrow^* \Sigma' = \Phi'; \Psi'; \mathcal{C}'; \mathcal{N}'; \sigma'$ *faite de règles de pré-résolution, on a*

$$\begin{array}{rcl} |\Sigma|^1_{\text{st}} & \geq & |\Sigma'|^1_{\text{st}} \\ \text{narr}(\Sigma) & \geq & \text{narr}(\Sigma') \\ \text{c}(\Sigma) & = & \text{c}(\Sigma') \\ \text{nres}(\Sigma) & \geq & \text{nres}(\Sigma') \end{array}$$

De plus, si $\text{narr}(\Sigma) = \text{narr}(\Sigma')$*, alors*

$$\begin{array}{rcl} \ell(\Sigma) & \geq & \ell(\Sigma') \\ \text{nmatch}(\Sigma) & \geq & \text{nmatch}(\Sigma') \end{array}$$

Enfin si $\text{nres}(\Sigma) = \text{nres}(\Sigma')$*, alors* $\sigma'|_{\mathcal{X}^1} = \sigma|_{\mathcal{X}^1}$ *et tous les* mgu *calculés par la règle* **Project** *lors de la dérivation valent l'identité.*

Démonstration. Il suffit de montrer la propriété pour une seule étape de règle.

Le seul cas non évident pour ce qui concerne les règles **Imitate** et **Coalesce** est la propriété $\ell(\Sigma) \geq \ell(\Sigma')$: celle-ci découle du lemme 4.50.

Concernant la règle **Project**, $|\Sigma|^1_{\text{st}} \geq |\Sigma'|^1_{\text{st}}$ découle de la définition de l'unificateur principal μ.

L'inégalité $\text{narr}(\Sigma') \leq \text{narr}(\Sigma)$ découle des lemmes 4.49 et 4.47.

L'égalité $\text{c}(\Sigma) = \text{c}(\Sigma')$ est claire, tandis que $\text{nres}(\Sigma) \geq \text{nres}(\Sigma')$ est une conséquence de lemme 4.48 (μ étant bien idempotente et vérifiant $\text{var}(\mu) \subseteq \text{var}^1(\Phi, \mathcal{C})$ par construction). D'après ce même lemme, $\text{nres}(\Sigma) = \text{nres}(\Sigma')$ implique bien $\mu = \{\}$ soit $\sigma'|_{\mathcal{X}^1} = \sigma|_{\mathcal{X}^1}$.

Enfin, supposons que $\text{narr}(\Sigma') = \text{narr}(\Sigma)$. Par le lemme 4.47, étant donné la définition de la règle, on a alors $\Phi'^{\mathcal{N}'} \subseteq (\Phi^{\mathcal{N}})\mu = (\Phi^{\mathcal{N}})\sigma'$.

On en déduit que $\text{Nmatch}(\Sigma') \subseteq \{(t\mu, l') \mid (t, l') \in \text{Nmatch}(\Sigma)\}$ soit $\text{nmatch}(\Sigma') \leq \text{nmatch}(\Sigma)$.

De plus, les lemmes 4.49 et 4.47 s'appliquent alors pour donner $\ell(\Sigma') \leq \ell(\Sigma)$. □

Lemme 4.52. *Soit* $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ *un système de contraintes accessible. Pour toute dérivation* $\Sigma \Longrightarrow \Sigma' = \Phi'; \Psi'; \mathcal{C}'; \mathcal{N}'; \sigma'$ *par la règle* **Discard** *on a*

$$\begin{array}{rcl} |\Sigma|^1_{\text{st}} & \geq & |\Sigma'|^1_{\text{st}} \\ \text{narr}(\Sigma) & \geq & \text{narr}(\Sigma') \\ \text{c}(\Sigma) & = & \text{c}(\Sigma') \\ \text{nres}(\Sigma) & \geq & \text{nres}(\Sigma') \\ \text{sar}(\Sigma) & = & \text{sar}(\Sigma') \end{array} \qquad \begin{array}{rcl} \ell(\Sigma) & \geq & \ell(\Sigma') \\ \text{nmatch}(\Sigma) & \geq & \text{nmatch}(\Sigma') \\ \text{qadd}(\Sigma) & \geq & \text{qadd}(\Sigma') \\ \sigma' & = & \sigma \end{array}$$

Démonstration. La seule propriété non évidente au vu des définitions est $\ell(\Sigma) \geq \ell(\Sigma')$. Celle-ci découle du lemme 4.50. □

Dans ce qui suit, on appelle pré-résolu un ensemble de contraintes $\mathcal{C}_0$ de la forme $\mathcal{C}_0 = \{X_1 \rhd^? x_1, \ldots, X_n \rhd^? x_n\}$ où $n \geq 0$ et les variables $X_1, \ldots, X_n, x_1, \ldots, x_n$ sont deux à deux disjointes.

Lemme 4.53. *Soit $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}$; σ un système de contraintes, et posons $\mathcal{C} = \mathcal{C}_0 \uplus \mathcal{C}_1$ pour un certaine ensemble pré-résolu $\mathcal{C}_0$.*

Pour toute dérivation $\Sigma \Longrightarrow^* \Sigma' = \Phi'; \Psi'; \mathcal{C}'; \mathcal{N}'; \sigma'$ faite de règles de pré-résolution sans unification (i.e. tous les mgu *de* **Project** *sont l'identité), il existe une substitution (bien formée) ρ telle que*

(i) $\sigma' = \sigma\rho$,

(ii) $\mathrm{supp}(\rho) \subseteq \mathcal{X}^2$,

(iii) $\rho|_{\mathrm{var}(\Sigma) - \mathrm{var}^2(\mathcal{C}_1)}$ *est un renommage bien formé,*

(iv) $\mathcal{C}'$ *s'écrit* $\mathcal{C}' = \mathcal{C}'_0 \uplus \mathcal{C}'_1$ *où* $\mathcal{C}'_0 = \mathcal{C}_0\rho$ *est pré-résolu,* $\mathrm{var}^2(\mathcal{C}'_1) \cap \mathrm{var}(\Sigma) \subseteq \mathrm{var}^2(\mathcal{C}_1)$ *et* $\mathrm{var}^2(\mathcal{C}'_1) \cap \mathrm{var}((\mathrm{var}(\Sigma) - \mathrm{var}^2(\mathcal{C}_1))\rho) = \emptyset$,

(v) $\mathrm{var}^1(\mathcal{C}') \subseteq \mathrm{var}^1(\mathcal{C})$.

Démonstration. Nous prouvons les propriétés par récurrence sur la longueur de la dérivation. La dernière propriété $\mathrm{var}^1(\mathcal{C}') \subseteq \mathrm{var}^1(\mathcal{C})$ est claire par induction du fait que **Project** n'applique pas d'unificateur différent de $\{\}$.

Du fait de la contrainte de bonne formation syntaxique des systèmes de contraintes, notons que $\mathrm{var}^2(\mathcal{C}_0) \cap \mathrm{var}^2(\mathcal{C}_1) = \emptyset$.

Le cas de base $\Sigma = \Sigma'$ est clair. Soit $\Sigma \Longrightarrow^* \Sigma' = \Phi'; \Psi'; \mathcal{C}'; \mathcal{N}'; \sigma'$ une dérivation, ρ et μ vérifiant les conclusions du lemme. Soit $\mathcal{C}' = \mathcal{C}'_0 \uplus \mathcal{C}'_1$ avec $\mathcal{C}'_0 = \mathcal{C}_0\rho$. Par bonne formation de Σ', on a $\mathrm{var}^2(\mathcal{C}'_0) \cap \mathrm{var}^2(\mathcal{C}'_1) = \emptyset$.

Supposons que $\Sigma' \Longrightarrow \Sigma'' = \Phi''; \Psi''; \mathcal{C}''; \mathcal{N}''; \sigma''$ par une règle de pré-résolution sans unification. Nous distinguons trois cas selon la règle utilisée.

- **Project**. Comme $\mathcal{C}'_0$ est un sous-ensemble pré-résolu, la règle s'applique (par hypothèse sans unification) entre une contrainte $(X \rhd^? t) \in \mathcal{C}'_1 = \mathcal{C}' - \mathcal{C}'_0$ et une règle $(M \rhd t) \in \Phi' = \Phi\sigma'$.
  Comme $X \in \mathrm{var}(\mathcal{C}'_1)$, par le point (iv), on a $X \notin \mathrm{var}(\Sigma) - \mathrm{var}(\mathcal{C}_1)$ et $X \notin \mathrm{var}((\mathrm{var}^2(\Sigma) - \mathrm{var}^2(\mathcal{C}_1))\rho)$. Par conséquent, $\rho' = \rho\{X \mapsto M\}$ vérifie bien
  - $\sigma'' = \sigma\rho'$,
  - $\mathrm{supp}(\rho') \subseteq \mathcal{X}^2$,
  - $\rho'|_{\mathrm{var}(\Sigma) - \mathrm{var}^2(\mathcal{C}_1)} = \rho|_{\mathrm{var}(\Sigma) - \mathrm{var}^2(\mathcal{C}_1)}$ est un renommage bien formé,
  - $\mathcal{C}''$ s'écrit $\mathcal{C}'' = \mathcal{C}''_0 \uplus \mathcal{C}''_1$ où $\mathcal{C}''_0 = \mathcal{C}_0\rho' = \mathcal{C}_0\rho$ est pré-résolu, et $\mathcal{C}''_1 = \mathcal{C}'_1 - \{X \rhd^? t\}$ vérifie bien $\mathrm{var}^2(\mathcal{C}''_1) \cap \mathrm{var}^2(\Sigma) \subseteq \mathrm{var}^2(\mathcal{C}_1)$ et $\mathrm{var}^2(\mathcal{C}''_1) \cap \mathrm{var}((\mathrm{var}(\Sigma) - \mathrm{var}^2(\mathcal{C}_1))\rho') = \emptyset$.
- **Imitate**. Comme ci-dessus la variable substituée X vérifie $X \in \mathrm{var}^2(\mathcal{C}'_1)$; d'où $X \notin \mathrm{var}(\Sigma) - \mathrm{var}(\mathcal{C}_1)$ et $X \notin \mathrm{var}((\mathrm{var}^2(\Sigma) - \mathrm{var}^2(\mathcal{C}_1))\rho)$. Par conséquent, $\rho' = \rho\{X \mapsto f(X_1, \ldots, X_n)\}$ vérifie bien
  - $\sigma'' = \sigma\rho'$,
  - $\mathrm{supp}(\rho') \subseteq \mathcal{X}^2$,
  - $\rho'|_{\mathrm{var}(\Sigma) - \mathrm{var}^2(\mathcal{C}_1)} = \rho|_{\mathrm{var}(\Sigma) - \mathrm{var}^2(\mathcal{C}_1)}$ est un renommage bien formé,

– $\mathcal{C}''$ s'écrit $\mathcal{C}'' = \mathcal{C}_0'' \uplus \mathcal{C}_1''$ où $\mathcal{C}_0'' = \mathcal{C}_0\rho' = \mathcal{C}_0\rho$ est pré-résolu, et $\mathcal{C}_1'' = \mathcal{C}_1' - \{X \rhd^? t\} \cup \{X_1 \rhd^? t_1, \ldots, X_n \rhd^? t_n\}$ vérifie bien $\mathrm{var}^2(\mathcal{C}_1'') \cap \mathrm{var}^2(\Sigma) \subseteq \mathrm{var}^2(\mathcal{C}_1)$ car les $X_1, \ldots, X_n$ sont des variables fraîches, et $\mathrm{var}^2(\mathcal{C}_1'') \cap \mathrm{var}((\mathrm{var}(\Sigma) - \mathrm{var}^2(\mathcal{C}_1))\rho') = \emptyset$.

– **Coalesce**. Soit X_1 et X_2 les deux variables mises en jeu : $\mathrm{ar}(X_1) \leq \mathrm{ar}(X_2)$, le remplacement appliqué étant $\{X_2 \mapsto X_1\}$.
Posons $\rho' = \rho\{X_2 \mapsto X_1\}$. On a bien $\sigma'' = \sigma\rho'$, et $\mathrm{supp}(\rho') \subseteq \mathcal{X}^2$.
De plus, comme $\mathcal{C}_0'$ est un sous-ensemble pré-résolu, d'après la forme de la règle, X_1 et X_2 ne peuvent être tous les deux dans $\mathcal{C}_0'$.
Si $X_2 \in \mathrm{var}(\mathcal{C}_1')$, alors on a comme précédemment $X_2 \notin \mathrm{var}(\Sigma) - \mathrm{var}(\mathcal{C}_1)$, $X_2 \notin \mathrm{var}((\mathrm{var}^2(\Sigma) - \mathrm{var}^2(\mathcal{C}_1))\rho)$, et
– $\rho'|_{\mathrm{var}(\Sigma)-\mathrm{var}^2(\mathcal{C}_1)} = \rho|_{\mathrm{var}(\Sigma)-\mathrm{var}^2(\mathcal{C}_1)}$ est un renommage bien formé,
– $\mathcal{C}''$ s'écrit $\mathcal{C}'' = \mathcal{C}_0'' \uplus \mathcal{C}_1''$ où $\mathcal{C}_0'' = \mathcal{C}_0\rho' = \mathcal{C}_0\rho$ est pré-résolu, et $\mathcal{C}_1'' = \mathcal{C}_1' - \{X_2 \rhd^? t\}$ vérifie bien $\mathrm{var}^2(\mathcal{C}_1'') \cap \mathrm{var}^2(\Sigma) \subseteq \mathrm{var}^2(\mathcal{C}_1)$ et $\mathrm{var}^2(\mathcal{C}_1'') \cap \mathrm{var}((\mathrm{var}(\Sigma) - \mathrm{var}^2(\mathcal{C}_1))\rho') = \emptyset$.
Dans le cas contraire, on a $X_2 \in \mathrm{var}(\mathcal{C}_0')$ et $X_1 \in \mathcal{C}_1'$, de sorte que $X_1 \notin \mathrm{var}(\Sigma) - \mathrm{var}(\mathcal{C}_1)$, $X_1 \notin \mathrm{var}((\mathrm{var}^2(\Sigma) - \mathrm{var}^2(\mathcal{C}_1))\rho)$, et
– $\rho'|_{\mathrm{var}(\Sigma)-\mathrm{var}^2(\mathcal{C}_1)} = \rho|_{\mathrm{var}(\Sigma)-\mathrm{var}^2(\mathcal{C}_1)}\{X_2 \mapsto X_1\}$ est un renommage bien formé,
– $\mathcal{C}''$ s'écrit $\mathcal{C}'' = \mathcal{C}_0'' \uplus \mathcal{C}_1''$ où $\mathcal{C}_0'' = \mathcal{C}_0\rho' = \mathcal{C}_0\rho\{X_2 \mapsto X_1\}$ est pré-résolu, et $\mathcal{C}_1'' = \mathcal{C}_1' - \{X_1 \rhd^? t\}$ vérifie $\mathrm{var}^2(\mathcal{C}_1'') \cap \mathrm{var}^2(\Sigma) \subseteq \mathrm{var}^2(\mathcal{C}_1)$ et

$$\begin{aligned}&\mathrm{var}^2(\mathcal{C}_1'') \cap \mathrm{var}((\mathrm{var}(\Sigma) - \mathrm{var}^2(\mathcal{C}_1))\rho')\\ &\quad\subseteq (\mathrm{var}^2(\mathcal{C}_1') - \{X_1\}) \cap \mathrm{var}((\mathrm{var}(\Sigma) - (\mathrm{var}^2(\mathcal{C}_1))\rho) \cup \{X_1\}) = \emptyset\end{aligned}$$

□

4.5.4 Règles principales

Nous étudions pour finir la terminaison des règles principales non prioritaires, à savoir **Narrowing**, **Constrain**, **Context-{1,2}**, **Relate**, à l'aide de la mesure définie plus haut.

Lemme 4.54. *Soit Σ un système accessible par une dérivation standard et saturé par les règles prioritaires. Soit $\Sigma \Longrightarrow^+ \Sigma'$ une dérivation respectant les priorités entre règles et telle que tout système intermédiaire Σ'', $\Sigma \not\geq \Sigma''$.*

Supposons que Σ' est le premier système après Σ à être saturé par les règles prioritaires. Alors on a $\Sigma >_t \Sigma'$.

De plus, si les mesures $m_t(\Sigma)$ et $m_t(\Sigma')$ coïncident sur leurs cinq premières composantes alors $|\Sigma'|^1_{\mathrm{st}} \leq |\Sigma|^1_{\mathrm{st}}$.

Dans tous les autres cas, $|\Sigma'|^1_{\mathrm{st}} \leq |\Sigma|^1_{\mathrm{st}} + |\mathcal{R}|_{\mathrm{st}}$.

Démonstration. On distingue plusieurs cas selon la première règle de transformation **R** appliquée à Σ : $\Sigma \Longrightarrow_{\mathbf{R}} \Sigma_1 \Longrightarrow^* \Sigma'$.

Narrowing. Posons $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ et, avec les notations de la règle,

$$\begin{aligned}\Sigma_1 = \Phi_1;\ \Psi_1;\ \mathcal{C}_1;\ \mathcal{N}_1;\ \sigma_1\\ = (\Phi\{t_0 \mapsto r\};\ \Psi;\ \mathcal{C}\{t_0 \mapsto r\};\ \mathcal{N} \cup \{t_1, \ldots, t_n\};\ \sigma)\mu\end{aligned}$$

où $t_0 = f(t_1, \ldots, t_n) \in \mathrm{st}^1(\Phi, \mathcal{C}) - \mathcal{X} - \mathrm{st}(\mathcal{N})$ et $t_0\mu = l\mu$. En particulier, on a $t_0 \in \mathrm{Narr}(\Sigma) \stackrel{\Delta}{=} \mathrm{st}^1(\Phi, \mathcal{C}) - \mathrm{st}(\mathcal{N}) - \mathcal{F}[\emptyset]\downarrow_{\mathcal{R}}$.

Montrons que $\mathrm{Narr}(\Sigma_1) \subseteq (\mathrm{Narr}(\Sigma)\{t_0 \mapsto r\}\mu) - \{r\mu\}$, de sorte que $\mathrm{narr}(\Sigma_1) \leq \mathrm{narr}(\Sigma) - 1$.

Soit $t_1 \in \mathrm{Narr}(\Sigma_1) = \mathrm{st}^1(\Phi_1, \mathcal{C}_1) - \mathrm{st}(\mathcal{N}_1) - \mathcal{F}[\emptyset]\downarrow_{\mathcal{R}}$. Comme $\mathrm{supp}(\mu)\mu \subseteq \mathrm{st}_{\neq}(f(t_1, \ldots, t_n))\mu \subseteq \mathcal{N}_1$ (car $l \notin \mathcal{X}$), il existe $t \in \mathrm{st}^1(\Phi, \mathcal{C})$ tel que $t_1 = t\{t_0 \mapsto r\}\mu$.

Si $t \in \mathrm{st}(\mathcal{N}) \cup \mathcal{F}[\emptyset]\downarrow_{\mathcal{R}}$ alors comme $t_0 \notin \mathrm{st}(\mathcal{N}) \cup \mathcal{F}[\emptyset]\downarrow_{\mathcal{R}}$, on a $t_0 \notin \mathrm{st}(t)$ et donc $t_1 = t\mu \subseteq \mathrm{st}(\mathcal{N}_1) \cup \mathcal{F}[\emptyset]\downarrow_{\mathcal{R}}$ ce qui contredit $t_1 \in \mathrm{Narr}(\Sigma_1)$. Par conséquent, $t \in \mathrm{Narr}(\Sigma)$.

De plus, comme $r \in \mathrm{st}_{\neq}(l) \cup \mathcal{R}^{\emptyset}$ et $f(t_1, \ldots, t_n)\mu = l\mu$, on a $r\mu \in \mathcal{N}_1 \cup \mathcal{R}^{\emptyset}$; donc $t_1 \neq r\mu$.

En outre, comme par convention $\mathrm{var}(\mathrm{mgu}(t_0, l)) \subseteq \mathrm{var}(t_0, l)$, on a $|\Sigma|^1_{\mathrm{st}} \leq |\Sigma|_{\mathrm{st}} + |\mathcal{R}|_{\mathrm{st}}$.

Finalement, par les lemmes 4.51 et 4.52, on obtient $\Sigma >_t \Sigma'$ (avec en particulier $\mathrm{narr}(\Sigma) > \mathrm{narr}(\Sigma')$) et $|\Sigma'|^1_{\mathrm{st}} \leq |\Sigma|_{\mathrm{st}} + |\mathcal{R}|_{\mathrm{st}}$.

Constrain. Posons $\Sigma = \Phi;\ \Psi;\ \mathcal{C} \uplus \{t =^?_E t'\};\ \mathcal{N};\ \sigma$ et, avec les notations de la règle,

$$\Sigma_1 = (\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N} \cup \{t\};\ \sigma)\mu$$

Par le lemme 4.47, on a $\mathrm{Narr}(\Sigma_1) \subseteq \mathrm{Narr}(\Sigma\mu) \subseteq \mathrm{Narr}(\Sigma)\mu$.

Or, clairement $\mathrm{c}(\Sigma_1) < \mathrm{c}(\Sigma)$, et par ailleurs, du fait de la convention sur les mgu, on a $|\Sigma|^1_{\mathrm{st}} \leq |\Sigma|_{\mathrm{st}}$.

Finalement, par les lemmes 4.51 et 4.52, on obtient $\Sigma >_t \Sigma'$ et $|\Sigma'|^1_{\mathrm{st}} \leq |\Sigma|_{\mathrm{st}}$.

Relate. Avec les notations de la règle, posons $\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma$ et $\Sigma_1 = \Phi;\ \Psi \cup \{X \bowtie M\};\ \mathcal{C} \cup \{X \rhd^? t\};\ \mathcal{N};\ \sigma$.

On a clairement

- $\mathrm{Narr}(\Sigma_1) = \mathrm{Narr}(\Sigma)$,
- $\mathrm{c}(\Sigma_1) = \mathrm{c}(\Sigma)$,
- $\mathrm{Nmatch}(\Sigma_1) = \mathrm{Nmatch}(\Sigma)$,
- $\mathrm{Nres}(\Sigma_1) = \mathrm{Nres}(\Sigma)$,
- $|\Sigma_1|^1_{\mathrm{st}} = |\Sigma|^1_{\mathrm{st}}$.

De plus, $\ell(\Sigma_1) \leq \ell(\Sigma)$ découle des lemmes 4.49 et 4.50. Par les lemmes 4.51 et 4.52, on a donc

- $\mathrm{Narr}(\Sigma') \leq \mathrm{Narr}(\Sigma)$,
- $\mathrm{c}(\Sigma') \leq \mathrm{c}(\Sigma)$,
- $\mathrm{Nmatch}(\Sigma') \leq \mathrm{Nmatch}(\Sigma)$,
- $\mathrm{Nres}(\Sigma') \leq \mathrm{Nres}(\Sigma)$,
- $\ell(\Sigma') \leq \ell(\Sigma)$,
- $|\Sigma'|^1_{\mathrm{st}} \leq |\Sigma|^1_{\mathrm{st}}$.

Si l'une des cinq premières inégalités ci-dessus est stricte, alors $\Sigma >_t \Sigma'$.

Dans le cas contraire, soit $\Sigma_2 = \Phi_2; \Psi_2; \mathcal{C}_2; \mathcal{N}_2; \sigma_2$ le premier système pré-résolu après Σ_1. Comme on a en particulier $\mathrm{Nres}(\Sigma_2) = \mathrm{Nres}(\Sigma)$, d'après le lemme 4.51, tous les mgu calculés par la règle **Project** lors de la dérivation entre Σ_1 et Σ_2 valent l'identité.

Du fait que $\mathcal{C}$ est pré-résolu, en appliquant le lemme 4.53 à sous-dérivation

$$\Sigma_1 = \Phi;\ \Psi \cup \{X \bowtie M\};\ \mathcal{C} \cup \{X \rhd^? t\};\ \mathcal{N};\ \sigma \Longrightarrow^* \Sigma_2$$

on obtient qu'il existe une substitution bien formé ρ telle que

– $\sigma_2 = \sigma\rho$,
– $\mathrm{supp}(\rho) \subseteq \mathcal{X}^2$,
– $\rho|_{\mathrm{var}(\Sigma_1)-\{X\}} = \rho|_{\mathrm{var}(\Sigma)}$ est un renommage bien formé,
– $\mathcal{C}_2$ s'écrit $\mathcal{C}_2 = \mathcal{C}'_0 \uplus \mathcal{C}'_1$ où $\mathcal{C}'_0 = \mathcal{C}\rho$ est pré-résolu,
– $\mathrm{var}^1(\mathcal{C}_2) \subseteq \mathrm{var}^1(\mathcal{C} \cup \{X \rhd^? t\})$.

Or, sachant que $(M \rhd t) \in \Phi$, par stratification de Σ, on $\mathrm{var}(t) \subseteq \mathrm{var}^1(\mathcal{C})$. Le dernier point implique donc $\mathrm{var}^1(\mathcal{C}) = \mathrm{var}^1(\mathcal{C}'_0) \subseteq \mathrm{var}^1(\mathcal{C}_2) \subseteq \mathrm{var}^1(\mathcal{C})$. Autrement dit, comme $\mathcal{C}_2 = \mathcal{C}'_0 \uplus \mathcal{C}'_1$ est pré-résolu, on a $\mathrm{card}(\mathcal{C}_2) = \mathrm{card}(\mathcal{C}'_0)$, donc $\mathcal{C}'_1 = \emptyset$ et $\mathcal{C}_2 = \mathcal{C}\rho$. On obtient ainsi

$$\Sigma_2 = \Phi\rho;\ \Psi\rho \cup \{X\rho \bowtie M\rho\};\ \mathcal{C}\rho;\ \mathcal{N}\rho;\ \sigma\rho$$

Comme $\rho|_{\mathrm{var}(\Sigma)}$ est un renommage, on en déduit $\mathrm{qadd}(\Sigma_2) \leq \mathrm{qadd}(\Sigma)$. Si l'inégalité stricte, alors le lemme 4.52 implique $\mathrm{qadd}(\Sigma') \leq \mathrm{qadd}(\Sigma_2) < \mathrm{qadd}(\Sigma)$ et $\Sigma >_t \Sigma'$.

Dans le cas contraire, $\mathrm{qadd}(\Sigma_2) = \mathrm{qadd}(\Sigma)$ implique $(X\rho \bowtie M\rho) \in \Psi\rho$, d'où

$$\Sigma_2 = \Phi\rho;\ \Psi\rho;\ \mathcal{C}\rho;\ \mathcal{N}\rho;\ \sigma\rho$$

Sachant que $\mathrm{var}^*(\Sigma) \subseteq \mathrm{var}(\Sigma)$, si le renommage $\rho|_{\mathrm{var}^*(\Sigma)}$ est bien formé inversible, alors $\Sigma \geq \Sigma_2$; contradiction. Sinon, il existe $Y \in \mathrm{var}^*(\Sigma)$ tel que $\mathrm{ar}(Y\rho) < \mathrm{ar}(Y)$. Par conséquent, en utilisant le lemme 4.52

$$\mathrm{sar}(\Sigma') \leq \mathrm{sar}(\Sigma_2) < \mathrm{sar}(\Sigma)$$

et finalement $\Sigma >_t \Sigma'$.

Context-1. Avec les notations de la règle, la première transition s'écrit :

$$\begin{aligned}\Sigma = \Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N};\ \sigma \Longrightarrow \Sigma_1 = (\Phi \cup \{M' \rhd r\};\ \Psi \cup \{\forall Z_1, \ldots, Z_q.M \bowtie M'\};\\ \mathcal{C} \cup \{Y_1 \rhd^? y_1, \ldots, Y_p \rhd^? y_p\};\ \mathcal{N} \cup \{l_1, \ldots, l_n\};\ \sigma)\mu\\ = \Phi_1;\ \Psi_1;\ \mathcal{C}_1;\ \mathcal{N}_1;\ \sigma_1\end{aligned}$$

On a naturellement $\mathrm{c}(\Sigma_1) = \mathrm{c}(\Sigma)$. De plus, $|\Sigma_1|^1_{\mathrm{st}} \leq |\Sigma|_{\mathrm{st}} + |\mathcal{R}|_{\mathrm{st}}$.

Montrons que $\mathrm{narr}(\Sigma) \geq \mathrm{narr}(\Sigma_1)$. En effet, posons

$$\Sigma'_1 = (\Phi;\ \Psi;\ \mathcal{C};\ \mathcal{N} \cup \{l_1, \ldots, l_n\};\ \sigma)\mu$$

Par le lemme 4.47, on a $\mathrm{narr}(\Sigma'_1) \leq \mathrm{narr}(\Sigma)$.

Or, par définition de $\mu = \mathrm{mgu}((t_1, \ldots, t_n), (l_1, \ldots, l_n))$ et de $y_1, \ldots, y_n \in \mathrm{var}(t_1, \ldots, t_n)$, on a pour tout i, $y_i\mu \in \mathrm{st}(l_1\mu, \ldots, l_n\mu) \subseteq \mathrm{st}(\mathcal{N}_1)$. Pour la même raison, du fait de $r \in \mathrm{st}(l_1, \ldots, l_n) \cup \mathcal{F}[\emptyset]\downarrow_{\mathcal{R}}$, on a $r\mu \in \mathrm{st}(\mathcal{N}_1) \cup \mathcal{F}[\emptyset]\downarrow_{\mathcal{R}}$. Par conséquent, $\mathrm{narr}(\Sigma_1) = \mathrm{narr}(\Sigma'_1) \leq \mathrm{narr}(\Sigma)$.

Dans le cas où $\mathrm{narr}(\Sigma_1) < \mathrm{narr}(\Sigma)$, on conclut directement par les lemmes 4.51 et 4.52.

Supposons que l'on ait l'égalité $\mathrm{narr}(\Sigma_1) = \mathrm{narr}(\Sigma)$. Alors par le lemme 4.47, on a $(\Phi\mu)^{(\mathcal{N}_1)} \subseteq (\Phi^{\mathcal{N}})\mu$.

Notamment, les conditions du lemme 4.49 étant remplies entre Σ et Σ'_1, on a $\mathrm{Secr}(\Sigma'_1) \subseteq \mathrm{Secr}(\Sigma)\mu$.

Par ailleurs, $\Phi_1^{\mathcal{N}_1} = (\Phi\mu)^{\mathcal{N}_1} \cup \{M' \rhd r\mu\}$, d'où $\mathrm{Secr}(\Sigma_1) \subseteq \mathrm{Secr}(\Sigma'_1) \cup \mathrm{st}_{\neq}(r\mu)$.

Montrons que $\mathrm{Secr}(\Sigma_1) \subseteq \mathrm{Secr}(\Sigma_1')$. En effet, comme $r \in \mathrm{st}(l_1, \ldots, l_n) \cup \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}}$ et $l_i\mu = t_i\mu$, on a d'une part

$$r\mu \in \mathrm{st}(t_1\mu, \ldots, t_n\mu) \cup \mathcal{F}[\emptyset]{\downarrow_{\mathcal{R}}}$$

et d'autre part, $(M_i \rhd t_i) \in \Phi$ et $t_i\mu \in \mathrm{st}(\mathcal{N}_1)$ implique $(M_i \rhd t_i\mu) \in (\Phi\mu)^{\mathcal{N}_1}$.

Sachant que pour tout i, $M_i \preceq M'$, on en déduit $\mathrm{st}_{\neq}(r\mu) \cap \mathrm{Secr}(\Sigma_1) \subseteq \mathrm{Secr}(\Sigma_1')$, d'où $\mathrm{Secr}(\Sigma_1) \subseteq \mathrm{Secr}(\Sigma_1') \subseteq \mathrm{Secr}(\Sigma)$.

Soit $\Sigma_2 = \Phi_2; \Psi_2; \mathcal{C}_2; \mathcal{N}_2; \sigma_2$ le premier système pré-résolu après Σ_1 dans la dérivation considérée. Supposons sans perte de généralité que $\mathrm{narr}(\Sigma_2) = \mathrm{narr}(\Sigma_1)$. D'après le lemme 4.49 et ce qui précède, on a

$$\mathrm{Secr}(\Sigma_2) \subseteq \mathrm{Secr}(\Sigma_1)\sigma_2 \subseteq \mathrm{Secr}(\Sigma)\sigma_2$$

D'après le lemme 4.50 et le lemme 4.22, on a donc

$$\ell(\Sigma_2) \leq \ell(\Sigma_1) \leq \ell(\Sigma)$$

Supposons à nouveau sans perte de généralité que $\ell(\Sigma_2) = \ell(\Sigma)$. Étant donné le lemme 4.50, on en déduit : pour tout $t \in \mathrm{Secr}(\Sigma)$,

$$\ell(\Sigma_2, t\sigma_2) = \ell(\Sigma_1, t\mu) = \ell(\Sigma, t)$$

Montrons qu'il existe $N \preceq M'\sigma_2$ tel que $N \rhd_{\Phi\sigma_2, \mathcal{C}_2} r\sigma_2$.

En effet, si $r\sigma_2 \notin \mathrm{Secr}(\Sigma_2)$, alors comme

- $r\sigma_2 \in \mathrm{st}(t_1\sigma_2, \ldots, t_n\sigma_2)$,
- pour tout $1 \leq i \leq n$, $(M_i \rhd t_i\sigma_2) \in (\Phi\sigma_2)^{\mathcal{N}_2}$ et $M_i\sigma_2 \preceq M'\sigma_2$, et
- Σ_2 est pré-résolu,

par définition de $\mathrm{Secr}(\Sigma_2)$, on en déduit qu'il existe $N \preceq M_i \preceq M'\sigma_2$ tel que $N \rhd_{\Phi_2{}^{\mathcal{N}_2}, \mathcal{C}_2} t$.

Dans le cas contraire, $r\sigma_2 \in \mathrm{Secr}(\Sigma_2) \subseteq \mathrm{Secr}(\Sigma)\sigma_2$. Soit donc $t_0 \in \mathrm{Secr}(\Sigma)$ tel que $r\sigma_2 = t_0\sigma_2$. Par hypothèse, on a $\ell(\Sigma_2, r\sigma_2) = \ell(\Sigma, t_0)$.

D'autre part, comme le système Σ_2 pré-résolu et accessible par une dérivation respectant les priorités entre règles, la proposition 4.28 implique que les équations de Φ_2 sont totalement stratifiées. Soit $k' \triangleq \mathrm{maxpar}(M'\sigma_2)$. Comme $(M'\sigma_2 \rhd r\sigma_2) \in \Phi_2$, on a donc $\mathrm{maxar}(M'\sigma_2) < k'$ et $\ell(\Sigma_2, r\sigma_2) = \ell(\Sigma, t_0) \leq k'$.

Soit donc N_0 tel que $N_0 \rhd_{\Phi, \mathcal{C}} t_0$ et $\max(\mathrm{maxpar}(N_0), \mathrm{maxar}(N_0) + 1) \leq k'$ (avec la convention $-\infty + 1 = -\infty$).

Le corollaire 4.26 montre que pendant la phase règles de pré-résolution suivant Σ_1, la règle **Project** n'utilise pas la règle ajoutée $M' \rhd r\mu$; dans ces conditions, on déduit de la preuve du lemme 4.21 que $N_0 \rhd_{\Phi, \mathcal{C}} t_0$ implique $N_0\sigma_2 \rhd_{\Phi\sigma_2, \mathcal{C}_2} t_0\sigma_2$ ($\Phi\sigma_2$ au lieu de Φ_2).

Ainsi, en posant $N = N_0\sigma_2$, on a $N \rhd_{\Phi\sigma_2, \mathcal{C}_2} t_0\sigma_2 = r\sigma_2$ et, comme σ_2 est bien formée, $\max(\mathrm{maxpar}(N), \mathrm{maxar}(N) + 1) \leq k'$. Par conséquent, étant donné que $k' = \mathrm{maxpar}(M'\sigma_2)$, on a $N \preceq M'\sigma_2$.

Nous avons montré qu'il existait $N \preceq M'\sigma_2$ tel que $N \rhd_{\Phi\sigma_2, \mathcal{C}_2} r\sigma_2$. Par définition des dérivations standard, deux cas sont donc possibles :

- $(M'\sigma_2 \rhd r\sigma_2) \in \Phi\sigma_2$, ou bien
- la règle $M'\sigma_2 \rhd r\sigma_2$ est éliminée par **Discard** à la prochaine transition.

Dans les deux cas, il existe donc une sous-dérivation $\Sigma_2 \Longrightarrow^{\leq 1}_{\mathbf{Discard}} \Sigma_2'$ et

$$\Sigma_2' = \Phi\sigma_2;\ \Psi_2';\ \mathcal{C}_2;\ \mathcal{N}_2;\ \sigma_2$$

avec $\Psi_2 \subseteq \Psi_2' \subseteq \Psi_2 \cup \{N \bowtie M'\sigma_2\}$.

Comme nous avons supposé $\mathrm{narr}(\Sigma_2) = \mathrm{narr}(\Sigma_1) = \mathrm{narr}(\Sigma)$, on a

$$(\Phi\sigma_2)^{\mathcal{N}_2} \subseteq (\Phi_2)^{\mathcal{N}_2} \subseteq (\Phi^{\mathcal{N}})\sigma_2$$

Par conséquent,

$$\begin{array}{rcl} \mathrm{Nmatch}(\Sigma_2') & \subseteq & \{(t'\sigma_2, l')\ (t', l') \in \mathrm{Nmatch}(\Sigma\mu)\} \\ & \subseteq & \{(t\sigma_2, l')\ (t, l') \in \mathrm{Nmatch}(\Sigma)\} \end{array}$$

et en particulier $\mathrm{nmatch}(\Sigma_2') \leq \mathrm{nmatch}(\Sigma)$.

Si $\mathrm{nmatch}(\Sigma_2') < \mathrm{nmatch}(\Sigma)$, on conclut $\Sigma >_t \Sigma'$. Supposons l'égalité. Dans ce cas, étant donné la définition de μ, cela signifie que pour tout $1 \leq i \leq n$, t_i est une instance l_i. Nous avons supposé par convention que dans ce cas-là, $\mathrm{supp}(\mu) \subseteq \mathrm{var}(l_1, \ldots, l_n)$. Ainsi Σ_1 s'écrit

$$\begin{array}{l} \Sigma_1 = \Phi \cup \{M' \rhd r\mu\};\ \Psi \cup \{\forall Z_1, \ldots, Z_q.M \bowtie M'\}; \\ \qquad\qquad \mathcal{C} \cup \{Y_1 \rhd^? y_1\mu, \ldots, Y_p \rhd^? y_p\mu\};\ \mathcal{N} \cup \{l_1\mu, \ldots, l_n\mu\};\ \sigma\mu \end{array}$$

De plus, $\mathrm{var}(r\mu, y_1\mu, \ldots, y_n\mu) \subseteq \mathrm{var}(t_1, \ldots, t_n)$ et par conséquent $\mathrm{Nres}(\Sigma_1) \subseteq \mathrm{Nres}(\Sigma)$.

Les lemmes 4.51 et 4.52 impliquent donc $\mathrm{nres}(\Sigma_2') \leq \mathrm{nres}(\Sigma_2') \leq \mathrm{nres}(\Sigma_1) \leq \mathrm{nres}(\Sigma)$.

À nouveau, on peut supposer l'égalité sans perte de généralité. Dans ce cas, comme précédemment la pré-résolution s'effectue sans unification, et le lemme 4.53 s'applique donc à la dérivation

$$\begin{array}{rcl} \Sigma_1 & = & \Phi \cup \{M' \rhd r\mu\};\ \Psi \cup \{\forall Z_1, \ldots, Z_q.M \bowtie M'\}; \\ & & \quad \mathcal{C} \cup \{Y_1 \rhd^? y_1\mu, \ldots, Y_p \rhd^? y_p\mu\};\ \mathcal{N} \cup \{l_1\mu, \ldots, l_n\mu\};\ \sigma\mu \\ & = & \Phi_1;\ \Psi_1;\ \mathcal{C}_1;\ \mathcal{N}_1;\ \sigma_1 \\ & \Longrightarrow^* & \Sigma_2 \end{array}$$

pour donner un substitution ρ telle que

- $\sigma_2 = \sigma\rho$,
- $\mathrm{supp}(\rho) \subseteq \mathcal{X}^2$,
- $\rho|_{\mathrm{var}(\Sigma_1) - \{Y_1, \ldots, Y_n\}} = \rho|_{\mathrm{var}(\Sigma)}$ est un renommage bien formé,
- $\mathcal{C}_2$ s'écrit $\mathcal{C}_2 = \mathcal{C}_0' \uplus \mathcal{C}_1'$ où $\mathcal{C}_0' = \mathcal{C}\rho$ est pré-résolu,
- $\mathrm{var}^1(\mathcal{C}_2) \subseteq \mathrm{var}^1(\mathcal{C} \cup \{Y_1 \rhd^? y_1\mu, \ldots, Y_p \rhd^? y_p\mu\})$.

Par stratification de Σ et comme $\mathrm{var}(y_1\mu, \ldots, y_n\mu) \subseteq \mathrm{var}(t_1, \ldots, t_n)$, le deuxième point implique $\mathrm{var}^1(\mathcal{C}) = \mathrm{var}^1(\mathcal{C}_0') \subseteq \mathrm{var}^1(\mathcal{C}_2) \subseteq \mathrm{var}^1(\mathcal{C})$. Autrement dit, comme $\mathcal{C}_2$ est pré-résolu, $\mathcal{C}_1' = \emptyset$ ou encore $\mathcal{C}_2 = \mathcal{C}\rho$. On obtient ainsi

$$\begin{array}{l} \Sigma_2 = (\Phi \cup \{M' \rhd r\mu\})\rho;\ \Psi\rho \cup \{(\forall Z_1, \ldots, Z_q.M \bowtie M')\rho\}; \\ \qquad\qquad\qquad\qquad\qquad\qquad \mathcal{C}\rho;\ \mathcal{N} \cup \{\{l_1\mu, \ldots, l_n\mu\}\rho;\ \sigma\rho \end{array}$$

et par conséquent

$$\Sigma_2' = \Phi\rho;\ \Psi_2';\ \mathcal{C}\rho;\ \mathcal{N} \cup \{\{l_1\mu, \ldots, l_n\mu\}\rho;\ \sigma\rho$$

où

$$\Psi_2 \cup \{(\forall Z_1, \ldots, Z_q.M \bowtie M')\rho\} \subseteq \Psi_2' \subseteq \Psi_2 \cup \{(\forall Z_1, \ldots, Z_q.M \bowtie M')\rho,\ N \bowtie M'\rho\}$$

Comme $\rho|_{\mathrm{var}(\Sigma)}$ est un renommage, on en déduit $|\Sigma'|^1_{\mathrm{st}} \leq |\Sigma|^2_{\mathrm{st}} \leq |\Sigma|^1_{\mathrm{st}}$ et $\mathrm{qadd}(\Sigma_2') \leq \mathrm{qadd}(\Sigma)$.

Si la dernière inégalité est stricte, alors le lemme 4.52 implique $\mathrm{qadd}(\Sigma') \leq \mathrm{qadd}(\Sigma_2') < \mathrm{qadd}(\Sigma)$ et $\Sigma >_t \Sigma'$.

Dans le cas contraire, $\mathrm{qadd}(\Sigma_2') = \mathrm{qadd}(\Sigma)$ implique $(\forall Z_1, \ldots, Z_q.M \bowtie M')\rho \in \Psi\rho$ et (le cas échéant) $(N \bowtie M'\rho) \in \Psi\rho$, d'où finalement

$$\Sigma_2' = \Phi\rho;\ \Psi\rho;\ \mathcal{C}\rho;\ \mathcal{N}\rho;\ \sigma\rho$$

Sachant que $\mathrm{var}^*(\Sigma) \subseteq \mathrm{var}(\Sigma)$, si le renommage $\rho|_{\mathrm{var}^*(\Sigma)}$ est bien formé inversible, alors $\Sigma \geq \Sigma_2'$; contradiction. Sinon, il existe $Y \in \mathrm{var}^*(\Sigma)$ tel que $\mathrm{ar}(Y\rho) < \mathrm{ar}(Y)$. Par conséquent, en utilisant le lemme 4.52

$$\mathrm{sar}(\Sigma') \leq \mathrm{sar}(\Sigma_2') < \mathrm{sar}(\Sigma)$$

et finalement $\Sigma >_t \Sigma'$.

Context-2. Le cas de **Context-2** est similaire à celui de **Context-1** (la discussion sur $\ell(\Sigma')$ en moins). □

4.5.5 Conclusion de la preuve de terminaison

Nous pouvons désormais établir la terminaison de la procédure de décision décrite précédemment (sous-section 4.1.2 et suivantes), basée sur l'énumération des systèmes de contraintes accessibles par les dérivations standard.

Proposition 4.55 (Terminaison de la procédure)**.** *Étant donné un système de contraintes initial Σ_0, le nombre de systèmes de contraintes Σ (modulo renommage $\equiv$) tels que $\Sigma_0 \Longrightarrow^* \Sigma$ par une dérivation standard est fini.*

De plus, il existe une fonction polynomiale f ne dépendant que de $\mathcal{R}$ telle que pour tout Σ obtenu de cette manière, $|\Sigma|^1_{\mathrm{st}} \leq f(|\Sigma_0|^1_{\mathrm{st}})$.

Démonstration. À partir du lemme 4.54, sachant que les règles prioritaires terminent, et que l'ordre $>_t$ est bien fondé, on en déduit la terminaison des dérivations standard. Comme le nombre de successeurs d'un système (modulo renommage) par les règles est fini, on obtient par le lemme de König que le nombre de systèmes accessibles par de telles dérivations est fini.

Par ailleurs, le lemme 4.54 montre que seules les instances de règles principales faisant décroître les cinq premières composantes :

$$(\mathrm{narr}(\Sigma),\ \mathrm{c}(\Sigma),\ \ell(\Sigma),\ \mathrm{nmatch}(\Sigma), \mathrm{nres}(\Sigma))$$

sont susceptibles d'augmenter la taille au premier ordre $|\Sigma|^1_{\mathrm{st}}$ des systèmes Σ, ce d'au plus $|\mathcal{R}|_{\mathrm{st}}$.

Étant donné que, chacune de ces composantes est elle-même bornée polynomialement par $|\Sigma|^1_{\mathrm{st}}$, et que les règles prioritaires n'augmentent pas $|\Sigma|^1_{\mathrm{st}}$, on en déduit qu'une dérivation standard à partir de Σ_0 conduit à des systèmes accessibles de taille polynomiale au premier ordre dans $|\Sigma_0|^1_{\mathrm{st}}$. □

4.6 Obtention d'un algorithme NP

Les résultats obtenus (propositions 4.12, 4.30 et 4.55) fournissent d'ores et déjà un algorithme de décision pour la satisfiabilité et la S-équivalence (positive), consistant à parcourir les systèmes de contraintes accessibles par une dérivation standard à partir de chaque système initial, et à appliquer le critère du lemme 4.5.

Toutefois, la procédure décrite ne permet pas de conclure directement que le problème de la S-équivalence est dans co-NP. En effet, on remarque — notamment à travers la mesure $\mathrm{Qadd}(\Sigma)$ utilisée dans la preuve de terminaison — que l'ensemble des équations générées par les règles de transformation est de taille potentiellement exponentielle dans la taille du système initial.

Ce phénomène semble inévitable étant donnée la formulation des règles. Toutefois nous pouvons conclure la preuve du théorème 4.1 sans efforts supplémentaires importants à l'aide

- du critère fourni par la proposition 4.5 concernant les systèmes résolus,
- de la correction et de la complétude des règles de transformation (pour les dérivations standard),
- de l'analyse précise de la taille des systèmes générés au premier ordre, réalisée dans la section précédente,
- et de la procédure de M. Abadi et V. Cortier [AC04] permettant de décider la déductibilité et l'équivalence statique en temps polynomial pour les théories sous-termes-convergentes.

Pour commencer, nous rassemblons une partie des résultats de ce chapitre dans la proposition suivante.

Proposition 4.56. *On suppose la théorie E engendrée par un système de réécriture sous-terme-convergent et fini $\mathcal{R}$.*

1. *Soit Σ un système de contraintes initial. Σ est satisfiable ss'il existe θ et λ tels que*
   - *$\theta, \lambda \models^\sharp \Sigma$ (voir section 4.2 pour la notation $\models^\sharp$) ;*
   - *la taille DAG de $\lambda|_{\mathrm{var}^1(\Sigma)}$, c'est-à-dire $|\mathrm{var}^1(\Sigma)\,\lambda|_{\mathrm{st}}$, est bornée par un polynôme en $|\Sigma|$ (ne dépendant que de $\mathcal{R}$) ;*
   - *les termes dans l'image de $\lambda|_{\mathrm{var}^1(\Sigma)}$ utilisent seulement des symboles présents dans Σ et $\mathcal{R}$, ou des constantes libres publiques de sortes intervenant dans Σ et $\mathcal{R}$.*
2. *Soient $\Sigma_1 = \Phi_1; \emptyset; \mathcal{C}_1; , \emptyset; \{\}$ et $\Sigma_2 = \Phi_2; \emptyset; \mathcal{C}_2; , \emptyset; \{\}$ deux systèmes de contraintes initiaux et standard, Σ_2 étant de plus positif. Les solutions de Σ_1 ne sont pas incluses dans celles de Σ_2 ss'il existe θ et λ_1 tels que*
   - *$\theta, \lambda_1 \models^\sharp \Sigma_1$,*

- *la taille DAG de* $\lambda_1|_{\mathrm{var}^1(\Sigma)}$ *est bornée par un polynôme en* $|\Sigma_1|$ *(ne dépendant que de* $\mathcal{R}$*) ;*
- *les termes dans l'image de* $\lambda|_{\mathrm{var}^1(\Sigma)}$ *utilisent seulement des symboles présents dans* Σ_1 *et* $\mathcal{R}$*, ou des constantes libres publiques de sortes intervenant dans* Σ_1 *et* $\mathcal{R}$ *;*
- *l'une des conditions suivantes est vérifiée :*

  (a) $\theta \not\models \Sigma_2$,

  (b) l'unique λ_2 *tel que* $\theta, \lambda_2 \models \Sigma_2$ *est tel qu'il existe* M *et* N *vérifiant*

$$M[\Phi_1\lambda_1] =_E N[\Phi_1\lambda_1] \text{ et } M[\Phi_2\lambda_2] \neq_E N[\Phi_2\lambda_2]$$

- *enfin, tous* θ' *et* λ_1' *vérifiant* $\theta', \lambda_1' \models^\sharp \Sigma_1$ *et* $\lambda_1'|_{\mathrm{var}^1(\Sigma_1)} = \lambda_1|_{\mathrm{var}^1(\Sigma_1)}$*, satisfont l'alternative (a) ou (b) ci-dessus au même titre que* θ *et* λ_1*.*

Démonstration. **1.** D'après la propriété de complétude (proposition 4.30), si Σ est satisfiable alors il existe une dérivation standard allant de Σ vers un système résolu $\Sigma' = \Phi'; \Psi'; \mathcal{C}'; \mathcal{N}'; \sigma'$.

Par l'analyse de la terminaison (proposition 4.55), on a $|\Sigma'|^1_{\mathrm{st}} \leq f(|\Sigma|^1_{\mathrm{st}})$ pour une certain fonction polynomiale f ne dépendant que de $\mathcal{R}$. D'après la construction donnée dans la preuve de la proposition 4.5, on en déduit l'existence de θ_1 et λ_1 tels que $\theta_1, \lambda_1 \models \Sigma'$ et

$$|\mathrm{var}^1(\Sigma)\,\lambda_1|_{\mathrm{st}} = |\mathrm{var}^1(\Sigma)\,\sigma'|_{\mathrm{st}} \leq |\Sigma'|^1_{\mathrm{st}} \leq f(|\Sigma|^1_{\mathrm{st}})$$

Par la propriété de correction (proposition 4.12), on obtient $\theta_1, \lambda \models^\sharp \Sigma$ pour un certain λ tel que $\lambda|_{\mathrm{var}^1(\Sigma)} = \lambda_1|_{\mathrm{var}^1(\Sigma)}$.

Les conditions supplémentaires sur les symboles de $\lambda|_{\mathrm{var}^1(\Sigma)}$ découlent de la définition des règles et de la construction de $\lambda|_{\mathrm{var}^1(\Sigma)}$.

2. Supposons qu'il existe θ et λ_2 vérifiant les conditions données. Dans le cas (b), en utilisant la même construction que dans la preuve de proposition 4.5 (cas (2)⇒(1)) du fait que les systèmes sont standard, on en déduit l'existence de θ' tel que $\theta' \models \Sigma_1$ et $\theta' \not\models \Sigma_2$. Le cas (a) est évident.

Réciproquement, supposons qu'il existe θ_0 tel que $\theta_0 \models^\sharp \Sigma_1$ et $\theta_0 \not\models \Sigma_2$. Alors, d'après la propriété de complétude (proposition 4.30), il existe une dérivation standard allant de Σ_1 vers un système résolu $\Sigma = \Phi; \Psi; \mathcal{C}; \mathcal{N}; \sigma$ telle que $\theta_0 \models \Sigma$.

À nouveau par l'analyse de la terminaison (proposition 4.55), on a $|\Sigma|^1_{\mathrm{st}} \leq f(|\Sigma_1|^1_{\mathrm{st}})$. De plus, d'après la construction donnée dans la preuve de la proposition 4.5, on en déduit l'existence de θ et λ vérifiant les propriétés suivantes :

- $\theta, \lambda \models \Sigma$;
- $|\,\mathrm{var}^1(\Sigma_1)\,\lambda|_{\mathrm{st}} \leq f(|\Sigma_1|^1_{\mathrm{st}})$;
- si $\theta \models \Sigma_2$ alors il existe une équation $\forall\beta.M \bowtie N$ dans Ψ telle que

$$M[\Phi\lambda_2] \neq_E N[\Phi\lambda_2]$$

où λ_2 est l'unique substitution partielle telle que $\theta, \lambda_2 \models \Sigma$.

Par la propriété de correction (proposition 4.12), notamment concernant les équations de Ψ, on obtient donc pour un certain λ_1 tel que $\lambda_1|_{\mathrm{var}^1(\Sigma_1)} = \lambda|_{\mathrm{var}^1(\Sigma_1)}$:

- $\theta, \lambda_1 \models^\sharp \Sigma_1$;
- $|\,\mathrm{var}^1(\Sigma_1)\,\lambda_1|_{\mathrm{st}} \leq f(|\Sigma_1|^1_{\mathrm{st}})$;

– si $\theta_1 \models \Sigma_2$ alors il existe M et N tels que

$$M[\Phi_1\lambda_1] =_E N[\Phi_1\lambda_1] \text{ et } M[\Phi_2\lambda_2] \neq_E N[\Phi_2\lambda_2]$$

où λ_2 est l'unique substitution partielle telle que $\theta, \lambda_2 \models \Sigma_2$.

Pour finir nous vérifions le dernier point. Supposons que $\theta', \lambda'_1 \models^\sharp \Sigma_1$ et $\lambda_1|_{\mathrm{var}^1(\Sigma_1)} = \lambda'_1|_{\mathrm{var}^1(\Sigma_1)}$ pour un certain couple θ' et λ'_1.

Le système Σ_1 étant initial et standard, posons $\mathcal{C}_{1,\rhd} = \{X_1 \rhd^? x_1, \ldots, X_n \rhd^? x_n\}$ et $\Phi_1 = \{w_1 \rhd t_1, \ldots, w_{k_0} \rhd t_{k_0}\}$.

Comme Σ_1 est initial, les hypothèses $\theta, \lambda_1 \models^\sharp \Sigma_1$ et $\theta', \lambda'_1 \models^\sharp \Sigma_1$ impliquent en particulier :

– $\lambda_1|_\mathcal{Y} = \theta$,
– $\lambda'_1|_\mathcal{Y} = \theta'$,
– pour tout i, $(X_i\theta)[\Phi_1\lambda_1] =_E x_i\lambda_1$, et
– pour tout i, $(X_i\theta')[\Phi_1\lambda'_1] =_E x_i\lambda'_1$.

Or, par hypothèse, $\lambda'_1|_{\mathrm{var}^1(\Sigma_1)} = \lambda_1|_{\mathrm{var}^1(\Sigma_1)}$. Donc, pour tout i,

$$(X_i\theta)[\Phi_1\lambda_1] =_E (X_i\theta')[\Phi_1\lambda_1] \tag{4.6.1}$$

Compte tenu des conditions de régularité sur Σ_2, soit λ_2 et λ'_2 définis par induction sur i, par

– $\lambda_2|_\mathcal{Y} = \theta$,
– $\lambda'_2|_\mathcal{Y} = \theta'$,
– pour tout i, $(X_i\theta)[\Phi_2\lambda_2] = x_i\lambda_2$, et
– pour tout i, $(X_i\theta')[\Phi_1\lambda'_2] = x_i\lambda'_2$.

Par le fait 4.4, on sait que $\theta \models \Sigma_2$ ssi $\theta, \lambda_2 \models \Sigma_2$, et de même $\theta' \models \Sigma_2$ ssi $\theta', \lambda'_2 \models \Sigma_2$.

Supposons que θ' et λ'_1 ne vérifient pas la propriété (b) et montrons (a).

Comme (b) est faux et que l'équation 4.6.1 est valable dans $\Phi_1\lambda_1 = \Phi_1\lambda'_1$, on a pour tout i,

$$(X_i\theta)[\Phi_2\lambda'_2] =_E (X_i\theta')[\Phi_2\lambda'_2] = x_i\lambda'_2$$

Étant donné les conditions de régularité sur Σ_2, par induction sur i, on en déduit $x_i\lambda'_2 =_E x_i\lambda_2$.

Par conséquent, le point (b) est faux également pour θ et λ_1. Par hypothèse, (a) est donc vrai sur θ et λ_1. Or, du fait de l'équation ci-dessus, et Σ_2 étant initial, $\theta, \lambda_2 \models \Sigma_2$ implique $\theta', \lambda'_2 \models \Sigma_2$. □

Déductibilité et équivalence statique

Dans le formalisme de ce chapitre, les problèmes de déductibilité et d'équivalence statique (voir les chapitres 5 et 6) s'énoncent de la manière suivante :

Définition 4.15. Un *environnement* est un ensemble de règles

$$\Phi = \{w_1 \rhd t_1, \ldots, w_{k_0} \rhd t_{k_0}\}$$

tel que $\mathrm{var}(\Phi) = \emptyset$.

Un terme t (clos) est *déductible* de Φ ss'il existe un terme du second ordre M tel que $M[\Phi] =_E t$.

Deux environnements Φ_1 et Φ_2 sont *statiquement équivalents* ssi pour tous M et N, on a

$$M[\Phi_1] =_E N[\Phi_1] \Leftrightarrow M[\Phi_2] =_E N[\Phi_2]$$

Notons $\mathrm{eq}(\Phi)$ l'ensemble des équations visibles d'un environnement défini par

$$\mathrm{eq}(\Phi) \stackrel{\triangle}{=} \{(M, N) \mid M[\Phi] =_E N[\Phi]\}$$

Autrement dit, Φ_1 et Φ_2 sont statiquement équivalents ssi $\mathrm{eq}(\Phi_1) = \mathrm{eq}(\Phi_2)$.

Le résultat suivant est due à M. Abadi et V. Cortier [AC04, AC06].

Théorème 4.57. *Soit E une théorie engendrée par un système de réécriture sous-terme-convergent fini.*

La déductibilité, l'inclusion des ensembles d'équation et (donc) l'équivalence statique sont décidables en temps polynomial (dans la taille DAG des données du problème).

De plus, on peut calculer en temps polynomial un terme M de taille polynomiale vérifiant $M[\Phi] =_E t$ pour tout couple (Φ, t) tel que t est déductible de Φ.

Il convient de mentionner que la preuve de ce théorème est donnée initialement par M. Abadi et V. Cortier [AC04] dans un formalisme non typé, sans symboles privés excepté les noms, et pour des systèmes sous-terme-convergents dans le sens (légèrement plus strict) suivant :

> $\mathcal{R}$ est convergent, et pour tout $(l \to r) \in \mathcal{R}$, r est un sous-terme strict de l ou une constante publique (plutôt qu'un terme clos en forme normal).

Toutefois, les algorithmes proposés par M. Abadi et V. Cortier s'appliquent à notre formalisme sans modification substantielle.[3] Pour les besoins de l'analyse de complexité, nous admettons donc ce résultat tel qu'il est énoncé ici.

Conclusion de la preuve du théorème 4.1

À l'aide de la proposition 4.56 et du théorème ci-dessus, nous pouvons maintenant proposer un algorithme NP pour décider la satisfiabilité et la (non-)équivalence des systèmes de contraintes.

1. *Satisfiabilité.* Étant donné un système initial $\Sigma = \Phi; \emptyset; \mathcal{C}; \emptyset; \{\}$, on devine une substitution partielle close λ de domaine $\mathrm{var}^1(\Sigma)$, de taille DAG bornée polynomialement et de symboles choisis selon la proposition 4.56. (Il existe bien un nombre polynomial de tels Σ, modulo renommage des constantes libres publiques.) L'algorithme répond oui ssi les conditions suivantes sont réunies :
   - pour tout $X \rhd^? x$ dans $\mathcal{C}$, il existe M tel que $M[\Phi\lambda] =_E x\lambda$;
   - pour tout $t_1 =^?_E t_2$ dans $\mathcal{C}$, $t_1\lambda =_E t_2\lambda$;
   - pour tout $t_1 \neq^?_E t_2$ dans $\mathcal{C}$, $t_1\lambda \neq_E t_2\lambda$.

   (Ces conditions sont testables en temps polynomial d'après le théorème 4.57 et du fait que E est donnée par un système convergent fini.)

[3] On peut observer à titre indicatif que les sortes, les symboles privés et les membres droits de règle $l \to r$ non constants ne posent pas de difficultés particulières dans les preuves de ce chapitre.

2. *non-S-inclusion.* Étant donnés deux systèmes initiaux et standard Σ_1 et Σ_2, où Σ_2 est positif, on devine une substitution partielle close λ_1 de domaine $\mathrm{var}^1(\Sigma_1)$, de taille DAG bornée polynomialement et de symboles choisis selon la proposition 4.56.

   Posons $\Sigma_1 = \Phi_1; \emptyset; \mathcal{C}_1; \emptyset; \{\}$ et $\Sigma_2 = \Phi_2; \emptyset; \mathcal{C}_2; \emptyset; \{\}$, où

   $$\mathcal{C}_1 = \{X_1 \rhd^? x_1, \ldots, X_n \rhd^? x_n\}$$

   L'algorithme répond non si une des conditions suivantes n'est pas réunie :
   – pour tout $X_i \rhd^? x_i$ dans $\mathcal{C}_1$, il existe M_i tel que $M_i[\Phi_1\lambda_1] =_E x_i\lambda_1$;
   – pour tout $t_1 =^?_E t_2$ dans $\mathcal{C}_1$, $t_1\lambda_1 =_E t_2\lambda_1$;
   – pour tout $t_1 \neq^?_E t_2$ dans $\mathcal{C}_1$, $t_1\lambda_1 \neq_E t_2\lambda_1$.
   (Ces conditions sont testables en temps polynomial d'après le théorème 4.57. De plus, les M_i sont effectivement calculables.) Posons $X_i\theta = M_i$.

   Si $\theta \not\models \Sigma_2$, renvoyer oui. Dans le cas contraire, soit λ_2 tel que $\theta, \lambda_2 \models \Sigma_2$. (Ces opérations sont bien faisables en temps polynomial d'après la définition des systèmes de contraintes de l'intrus (chapitre 3) ou encore la preuve du fait 4.4.)

   Répondre oui ssi $\mathrm{eq}(\Phi_1\lambda_1)$ n'est pas inclus dans $\mathrm{eq}(\Phi_2\lambda_2)$. Ceci est possible en temps polynomial d'après le théorème 4.57.

3. *non-S-équivalence.* Étant donnés deux systèmes initiaux, standard et positifs Σ_1 et Σ_2, on teste à l'aide de la procédure NP ci-dessus s'il existe une solution de Σ_1 n'étant pas solution de Σ_2 ou inversement.

La correction et la complétude des trois procédures découlent de la proposition 4.56.

4.7 Conclusion et perspectives

Les parties précédentes constituent une preuve du résultat escompté (théorème 4.1) :

> La satisfiabilité et la non-S-équivalence positive des systèmes de contraintes de l'intrus sur une théorie sous-terme-convergente sont décidables en temps polynomial non-déterministe dans la taille des systèmes de contraintes.

Ceci permet de conclure l'étude de complexité des problèmes considérés dans le chapitre 3 dans le cas d'une théorie sous-terme-convergente : en particulier, les propriétés d'accessibilité considérées et la (non-)diff-équivalence de processus positifs finis constituent des problèmes NP-complets.

En comparaison avec les algorithmes de résolution de contraintes existants dans le domaine (par exemple [Hui99b, MS01, MS03, CKRT03a, DJ04a]), la contribution principale de ce chapitre est de prendre en compte non seulement la satisfiabilité mais aussi la S-équivalence des systèmes de contraintes de l'intrus.

Comme le suggèrent les travaux voisins de M. Abadi et V. Cortier dans le cas passif [AC04, AC06] ou encore ceux de B. Blanchet, M. Abadi et C. Fournet dans le cas de processus infinis (avec approximation) [Blab, BAF05], déterminer la S-équivalence entre systèmes de contraintes s'avère une tâche plus ardue que décider la satisfiabilité. En effet, il s'agit de caractériser les ensembles (généralement infinis) de solutions des systèmes de contraintes avec une précision suffisante pour pouvoir

les comparer. Cette précision se traduit par l'emploi de termes du second ordre pour représenter les calculs de l'attaquant, et explique d'après nous une grande partie de la difficulté du théorème 4.1.

En contrepartie, comme nous l'avons vu dans le chapitre précédent, se ramener à la S-équivalence fournit la première méthode pour décider la diff-équivalence de processus positifs finis — dont la résistance aux attaques par dictionnaire est un cas particulier. En outre, les travaux en cours de S. Delaune, S. Kremer et M. Ryan [Kre06] suggèrent que la S-équivalence s'applique également à la may-équivalence et (de façon approchée) à la congruence barbue entre processus finis du pi-calcul appliqué.

En ce qui concerne l'implémentation pratique de l'algorithme, nous disposons à l'heure actuelle d'un prototype [Bau] fonctionnant dans le cas passif, permettant de décider la déductibilité et l'équivalence statique pour une théorie sous-terme-convergente quelconque. Ce prototype, écrit en Ocaml, représente environ 1700 lignes de code.

Parmi les extensions théoriques envisageables, deux questions naturelles sont la généralisation du théorème 4.1 aux équivalences avec tests de différence, d'une part, et sa généralisation à une classe plus grande de théories équationnelles, d'autre part. Concernant le second point une manière élégante mais ambitieuse de procéder serait de généraliser le résultat de composition de Chevalier et Rusinowitch [CR05] aux problèmes d'équivalences de systèmes de l'intrus.

Deuxième partie

Justification cryptographique de modèles logiques

Chapitre 5

Comparaison des formalismes logiques et cryptographiques dans le cas passif

Sommaire

Nous avons vu dans le chapitre 3 comment spécifier la sécurité des protocoles cryptographiques d'un point de vue logique à l'aide d'un langage de processus paramétré par une certaine théorie équationnelle.

Décrire les propriétés algébriques des primitives de manière équationnelle permet de modéliser naturellement un grand nombre de primitives (voir chapitre 3). Lorsque les théories équationnelles s'y prêtent, les outils automatiques d'analyse fournissent alors un moyen efficace de rechercher les failles logiques d'un protocole.

Pour autant, en raison de leur niveau d'abstraction, les analyses logiques ne permettent pas a priori de déterminer la sécurité d'un protocole dans le modèle

cryptographique standard. Traduire l'absence de faille logique en une preuve de sécurité cryptographique est un problème difficile qui nécessite en particulier d'expliciter les hypothèses cryptographiques requises sur l'implémentation.

Les travaux conduits depuis plusieurs années dans ce domaine [AR02, BPW03a, BPW03b, BP04, MW04b, CW05, JLM05] ont montré que, dans de nombreux cas, une implémentation adéquate garantit que tout protocole logiquement sûr possède une preuve de sécurité cryptographique. Toutefois, ces résultats concernent des modèles abstraits non équationnels et un ensemble fixé de primitives : essentiellement les primitives usuelles (probabilistes) du chiffrement asymétrique, symétrique et de la signature asymétrique.

L'objet de ce chapitre est de présenter un ensemble de résultats et de techniques applicables à un jeu de primitives et à une théorie équationnelle arbitraires. Nous nous concentrons sur la notion d'*équivalence statique* provenant du pi-calcul appliqué [AF01]. Cette notion formelle modélise le fait qu'un attaquant ne puisse pas distinguer deux séquences de données — appelées aussi *environnements* (*frames*) — en exhibant une relation valable seulement sur l'une des deux séquences. Notamment, lorsque les données testées proviennent des messages d'un protocole, l'équivalence statique représente, pour une trace d'exécution fixée, le pouvoir de discernement d'un intrus passif.

Dans un premier temps (section 5.1), nous décrivons les formalismes abstraits et concrets utilisés ainsi que les notions correspondantes d'indistinguabilité.

Nous introduisons ensuite dans la section 5.2 les notions de sûreté et de fidélité d'une implémentation vis-à-vis de l'égalité, de la non-déductibilité et de l'équivalence statique. Concernant l'équivalence statique, la propriété de sûreté signifie notamment que les réalisations concrètes de deux environnements statiquement équivalents sont indistinguables par tout attaquant polynomial probabiliste. Inversement, la propriété de fidélité signifie que tout attaque logique est réalisable concrètement par un attaquant efficace (dans le même sens).

Après avoir étudié les relations entre ces différentes notions, nous montrons (section 5.3) que la sûreté d'une implémentation quelconque des groupes abéliens vis-à-vis de l'équivalence statique implique la difficulté du problème Diffie-Hellman décisionnel dans cette même implémentation (et de même pour le problème RSA après l'ajout d'un symbole libre à la signature).

Enfin, nous consacrons la section 5.4 à l'étude d'un critère suffisant de sûreté pour l'équivalence statique, intuitivement il s'agit que la sémantique concrète de tout environnement soit indistinguable d'une certaine sémantique idéale. L'étude de la complétude de ce critère nous conduit à introduire la notion d'environnements transparents et directement transparents, que nous voyons comme des outils complémentaires à la notion de motifs (*patterns*) de M. Abadi et de P. Rogaway [AR00].

5.1 Modèles abstraits, implémentations concrètes

5.1.1 Algèbres abstraites

Comme précédemment, nous représentons de manière abstraite les messages d'un protocole par une algèbre de termes du premier ordre $\mathcal{F}[\mathcal{X}]$ munie d'une théorie équationnelle E. Nous fixons une partition de $\mathcal{F}$ en symboles publics $\mathcal{F}_{\text{pub}}$, privés $\mathcal{F}_{\text{priv}}$ et un ensemble $\mathcal{N}$ de constantes libres dans E, appelées *noms*. Nous supposons

qu'il existe un nombre infini de noms de chaque sorte $\tau \in \mathcal{T}$. L'ensemble des noms apparaissant dans un terme T est noté $\mathrm{names}(T)$.

Toutefois, dans ce chapitre et dans les chapitres suivants, nous supposons que $\mathcal{F}_{\mathrm{priv}} = \emptyset$, autrement dit $\mathcal{F} = \mathcal{F}_{\mathrm{pub}} \uplus \mathcal{N}$. Ceci correspond au fait qu'il n'y a pas d'algorithme secret dans le modèle cryptographique usuel. Les valeurs secrètes sont ainsi générées exclusivement par des tirages au sort, modélisés par des noms.

Dans la suite, nous fixons un ensemble $\mathcal{W} = \bigcup_\tau \{w_1^\tau, w_2^\tau \ldots\}$ de variables spéciales appelées *paramètres*, vérifiant $\mathrm{sort}(w_i^\tau) = \tau$. Un *contexte public n-aire* est un terme C (non nécessairement linéaire) vérifiant $\mathrm{names}(C) = \emptyset$ et tel que $\mathrm{var}(C)$ est inclus dans $\{w_1^{\tau_1}, \ldots, w_n^{\tau_n}\}$ pour certains types $\tau_1, \ldots, \tau_n$. Si $T_1, \ldots, T_n$ sont des termes de sortes respectives $\tau_1, \ldots, \tau_n$, la notation $C[T_1, \ldots, T_n]$ signifie $C\{w_1^{\tau_1} \mapsto T_1, \ldots, w_n^{\tau_n} \mapsto T_n\}$.

5.1.2 Environnements, déductibilité et équivalence statique

La séquence des messages observés par l'attaquant pendant l'exécution d'un protocole est modélisée classiquement par un environnement (*frame* en anglais) [AF01, AC04].

Dans notre formalisme, un *environnement* φ est une substitution partielle close : $\varphi = \{x_1 \mapsto T_1, \ldots, x_n \mapsto T_n\}$ avec $\mathrm{dom}(\varphi) = \{x_1, \ldots, x_n\}$ et $\mathrm{var}(T_1, \ldots, T_n) = \emptyset$.

Du fait que l'on modélise un intrus passif, tous les noms apparaissant dans les environnements ou les termes d'un problème sont considérés comme ayant été générés par des agents honnêtes, et donc *a priori* secrets du point de vue de l'intrus.[1] Naturellement, il est toujours possible de divulguer explicitement un nom n en augmentant les environnements donnés à l'intrus par $x \mapsto n$, où x est une variable fraîche.

Ceci nous conduit à la définition suivante de la déductibilité d'un terme à partir d'un environnement.

Définition 5.1 (Déductibilité)**.** Un terme clos T est *déductible* à partir de φ dans la théorie E, ce que l'on note $\varphi \vdash_E T$, ss'il existe un terme M tel que $\mathrm{var}(M) \subseteq \mathrm{dom}(\varphi)$, $\mathrm{names}(M) \cap \mathrm{names}(\varphi, T) = \emptyset$ et $M\varphi =_E T$.

Prenons par exemple la théorie E_{enc} engendrée par $\mathsf{dec}(\mathsf{enc}(x, y), y) \doteq x$ et l'environnement $\varphi_1 = \{x_1 \mapsto \mathsf{enc}(k_1, k_2),\ x_2 \mapsto \mathsf{enc}(k_4, k_3),\ x_3 \mapsto k_3\}$ où $k_1, k_2, k_3, k_4 \in \mathcal{N}$: le nom k_4 est déductible de φ_1 puisque $\mathsf{dec}(x_2, x_3)\varphi_1 =_{E_{\mathsf{enc}}} k_4$ mais on peut montrer que ni k_1 ni k_2 ne sont déductibles.

La notion de déductibilité modélise le fait que l'intrus peut (ou non) récupérer la *totalité* d'un secret en observant les messages du protocoles. Afin de modéliser des connaissances partielles de l'intrus – par exemple le fait de pouvoir distinguer deux valeurs possibles du secret –, on introduit la notion d'équivalence statique.

Définition 5.2 (Équivalence statique)**.** Deux environnements φ_1 et φ_2 sont *statiquement équivalents* dans la théorie E, ce que l'on écrit $\varphi_1 \approx_E \varphi_2$, ssi

- $\mathrm{dom}(\varphi_1) = \mathrm{dom}(\varphi_2)$, et
- pour tous termes M et N tels que $\mathrm{var}(M, N) \subseteq \mathrm{dom}(\varphi_1)$ et $\mathrm{names}(M, N) \cap \mathrm{names}(\varphi_1, \varphi_2) = \emptyset$, on a l'équivalence

$$M\varphi_1 =_E N\varphi_1 \Leftrightarrow M\varphi_2 =_E N\varphi_2.$$

[1] Dans le formalisme du pi-calcul appliqué [AF01, AC04], on dirait que tous les noms du problème sont *liés* (ou *restreints*).

Par exemple, si $k \in \mathcal{N}$ et 0 et 1 sont deux constantes (publiques), on a dans la théorie précédente :

$$\begin{array}{rcl} \{x \mapsto \mathsf{enc}(0,k)\} & \approx_{E_{\mathsf{enc}}} & \{x \mapsto \mathsf{enc}(1,k)\} \\ \{x \mapsto \mathsf{enc}(0,k),\, y \mapsto k\} & \not\approx_{E_{\mathsf{enc}}} & \{x \mapsto \mathsf{enc}(1,k),\, y \mapsto k\} \end{array}$$

(Dans le second cas, considérer le test $\mathsf{dec}(x,y) \doteq 0$.) Notons que pour la seconde équivalence, l'ensemble des termes déductibles est le même de chaque côté.

5.1.3 Sémantique concrète

Nous décrivons maintenant la sémantique concrète des termes et des environnements, en fonction de l'implémentation choisie pour chaque primitive.

Définition 5.3. Étant donné une signature du premier ordre $(\mathcal{T}, \mathcal{F})$, une $(\mathcal{T}, \mathcal{F})$-*algèbre concrète* A (ou encore *algèbre calculatoire*, de l'anglais *computational algebra*) est la donnée :

- d'un ensemble non vide de chaînes de bits $[\![\tau]\!]_A \subseteq \{0,1\}^*$ pour chaque sorte $\tau \in \mathcal{T}$;
- d'une fonction calculable $[\![f]\!]_A : [\![\tau_1]\!]_A \times \cdots \times [\![\tau_k]\!]_A \to [\![\tau]\!]_A$ pour chaque $f \in \mathcal{F}_{\mathrm{pub}}$ tel que $\mathrm{ar}(f) = \tau_1 \times \cdots \times \tau_k \to \tau$;
- d'un congruence calculable $=_{A,\tau}$ pour chaque sorte τ, afin de tester l'égalité entre éléments de $[\![\tau]\!]_A$ (un même élément peut être représenté par deux chaînes de bits distinctes) ; par la suite, lorsque cela ne prête pas à confusion, nous écrivons $=_A$ plutôt que $=_{A,\tau}$;
- d'une procédure effective pour tirer au sort un élément $e \in [\![\tau]\!]_A$; nous écrivons $e \xleftarrow{R} [\![\tau]\!]_A$ un tel tirage ; nous supposons qu'aucune classe d'équivalence de $=_{A,\tau}$ n'a une probabilité 0 d'être tirée au sort.

Soit A une $(\mathcal{S}, \mathcal{F})$-algèbre concrète. On associe à chaque environnement $\varphi = \{x_1 \mapsto T_1, \ldots, x_n \mapsto T_n\}$ une distribution $\psi = [\![\varphi]\!]_A$, dont les tirages $\widehat{\psi} \xleftarrow{R} \psi$ sont calculés de la manière suivante :

1. pour tout nom $a \in \mathrm{names}(T_1, \ldots, T_n)$ de sorte τ, tirer une valeur $\widehat{a} \xleftarrow{R} [\![\tau]\!]_A$;
2. pour chaque x_i $(1 \le i \le n)$ de sorte τ_i, calculer $\widehat{T_i} \in [\![\tau_i]\!]_A$ récursivement sur la structure des termes : $\widehat{f(T'_1, \ldots, T'_m)} = [\![f]\!]_A(\widehat{T'_1}, \ldots, \widehat{T'_m})$;
3. renvoyer comme valeur l'expression $\widehat{\psi} = \{x_1 \mapsto \widehat{T_1}, \ldots, x_n \mapsto \widehat{T_n}\}$.

Les valeurs de la forme précédente $\phi = \{x_1 \mapsto e_1, \ldots, x_n \mapsto e_n\}$, où $e_i \in [\![\tau_i]\!]_A$, sont appelées *environnements concrets*.

Nous étendons la notation $[\![.]\!]_A$ aux (ensembles de) termes clos de manière évidente. Par exemple $e_1, e_2 \xleftarrow{R} [\![T_1, T_2]\!]_A$ désigne le calcul simultané d'une valeur de T_1 et d'une valeur de T_2, pour un même tirage aléatoire des valeurs de noms dans $\mathrm{names}(T_1, T_2)$.

Nous généralisons également la notation aux termes non clos, en spécifiant les valeurs (concrètes) de chaque variable : nous notons

$$[\![T]\!]_{A,\{x_1 \mapsto e_1, \ldots, x_n \mapsto e_n\}}$$

si $\mathrm{var}(T) = \{x_1, \ldots, x_n\}$. Notons que lorsqu'un terme ou un environnement ne contient pas de noms, la traduction précédente est déterministe. Dans ce cas, nous utilisons la même notation pour désigner une distribution et son unique valeur.

Dans la suite nous nous concentrons sur les notions de sécurité asymptotiques et supposons qu'une implémentation est la donnée d'une famille d'algèbres concrètes (A_η) indexées par un certain *paramètre de complexité* $\eta \geq 0$. Ce paramètre η peut être vu comme la taille des clés et autres valeurs secrètes. La *sémantique concrète* d'un environnement φ est ainsi une famille de distributions sur les environnements concrets $([\![\varphi]\!]_{A_\eta})$. Par la suite, nous considérons uniquement des familles d'algèbres concrètes (A_η) telles que chacune des opérations algébriques mentionnées dans la définition soient réalisables par un algorithme (uniforme) en temps polynomial dans η. Ceci garantit le fait que la sémantique concrète d'un environnement est calculable efficacement (dans le même sens).

On peut munir les familles de distributions sur les environnements concrets (ψ_η) de la notion cryptographique habituelle d'indistinguabilité. Intuitivement, deux familles (ψ_η) et (ψ'_η) sont *indistinguables*, ce que l'on note $(\psi_\eta) \approx (\psi'_\eta)$, ssi aucun adversaire en temps probabiliste polynomial $\mathcal{A}$ ne peut deviner s'il reçoit en entrée un échantillon de ψ_η ou de ψ'_η avec une probabilité significativement plus grande que $\frac{1}{2}$. Formellement, on demande que l'*avantage* de $\mathcal{A}$,

$$\mathrm{Adv}^{\mathrm{IND}}(\mathcal{A}, \eta, \psi_\eta, \psi'_\eta) = \\ \mathbb{P}[\widehat{\psi} \xleftarrow{R} \psi_\eta \ : \ \mathcal{A}(\eta, \widehat{\psi}) = 1] - \mathbb{P}[\widehat{\psi} \xleftarrow{R} \psi'_\eta \ : \ \mathcal{A}(\eta, \widehat{\psi}) = 1]$$

soit une fonction *négligeable* de η, c'est-à-dire soit inférieure à toute fonction η^{-p} $(p > 0)$ pour η assez grand.[2]

Une fonction $f(\eta)$ est dite *écrasante* (*overwhelming*) ssi $1 - f(\eta)$ est négligeable. Une famille de distributions (ψ_η) est *sans collision* (relativement à une famille de congruences $=_{A_\eta}$) ssi la probabilité de collision de deux tirages aléatoires de ψ_η, $\mathbb{P}[\, e_1, e_2 \xleftarrow{R} \psi_\eta \ : \ e_1 =_{A_\eta} e_2]$, est une fonction négligeable de η. Par une majoration classique, ceci équivaut à ce que la probabilité de tirer une certaine valeur e_0 (modulo $=_{A_\eta}$) par ψ_η, $\mathbb{P}\left[e \xleftarrow{R} \psi_\eta \ : \ e =_{A_\eta} e_0\right]$, soit bornée uniformément par un fonction négligeable de η.

Par convention, les adversaires considérés par la suite ont accès implicitement (par exemple via un oracle) à une source infinie de bits aléatoires, ainsi qu'au paramètre de complexité η.

5.2 Lien entre algèbres abstraites et concrètes

Dans la section précédente nous avons défini les notions d'algèbres abstraites et concrètes. Nous étudions maintenant le lien entre les notions formelles de l'égalité, la (non-)déductibilité et l'équivalence statique et leurs équivalents concrets, à savoir l'égalité, la sécurité à sens unique (*one-wayness*) et l'indistinguabilité.

5.2.1 Sûreté et fidélité d'une implémentation

Nous introduisons les notions de sûreté (*soundness*) et fidélité (*faithfulness*) des algèbres concrètes par rapport aux notions formelles étudiées : égalité, équivalence statique et déductibilité.

[2] Nous considérons ici les fonctions négatives comme négligeables.

Définition 5.4. Soit E une théorie équationnelle. Une famille d'algèbres concrètes (A_η) est

- $=_E$-*sûre* ssi pour tous termes clos T_1, T_2 de même sorte, $T_1 =_E T_2$ implique que $\mathbb{P}[\, e_1, e_2 \xleftarrow{R} [\![T_1,\, T_2]\!]_{A_\eta} \,:\, e_1 =_{A_\eta} e_2]$ est écrasante ;
- $=_E$-*fidèle* ssi pour tous termes clos T_1, T_2 de même sorte, $T_1 \neq_E T_2$ implique que $\mathbb{P}[\, e_1, e_2 \xleftarrow{R} [\![T_1,\, T_2]\!]_{A_\eta} \,:\, e_1 =_{A_\eta} e_2]$ est négligeable ;
- $\approx_E$-*sûre* ssi pour tous environnements φ_1, φ_2 de même domaine, $\varphi_1 \approx_E \varphi_2$ implique $([\![\varphi_1]\!]_{A_\eta}) \approx ([\![\varphi_2]\!]_{A_\eta})$;
- $\approx_E$-*fidèle* ssi pour tous environnements φ_1, φ_2 de même domaine, $\varphi_1 \not\approx_E \varphi_2$ implique l'existence d'un adversaire en temps polynomial $\mathcal{A}$ capable de distinguer efficacement les deux (familles d')environnements concrets, c'est-à-dire tel que $\mathrm{Adv}^{\mathrm{IND}}(\mathcal{A}, \eta, [\![\varphi_1]\!]_{A_\eta}, [\![\varphi_2]\!]_{A_\eta})$ est écrasant ;
- $\not\vdash_E$-*sûre* ssi pour tout environnement φ et terme clos T, $\varphi \not\vdash_E T$ implique que pour tout adversaire en temps polynomial $\mathcal{A}$, la fonction

$$\mathbb{P}[\phi, e \xleftarrow{R} [\![\varphi, T]\!]_{A_\eta} \,:\, \mathcal{A}(\phi) =_{A_\eta} e]$$

  est négligeable ;
- $\not\vdash_E$-*fidèle* ssi pour tout environnement φ et terme clos T, $\varphi \vdash_E T$ implique l'existence d'un adversaire en temps polynomial $\mathcal{A}$ tel que

$$\mathbb{P}[\phi, e \xleftarrow{R} [\![\varphi, T]\!]_{A_\eta} \,:\, \mathcal{A}(\phi) =_{A_\eta} e]$$

  est écrasante.

Parfois, il est possible d'établir des notions de sécurité plus fortes, sans restriction sur le temps de calcul de l'attaquant. Notamment, nous dirons que (A_η) est

- *inconditionnellement* $=_E$-*sûre* ssi pour tous termes clos T_1, T_2 de même sorte, $T_1 =_E T_2$ implique $\mathbb{P}[\, e_1, e_2 \xleftarrow{R} [\![T_1,\, T_2]\!]_{A_\eta} \,:\, e_1 =_{A_\eta} e_2] = 1$;
- *inconditionnellement* $\approx_E$-*sûre* ssi pour tous environnements φ_1, φ_2 de même domaine, $\varphi_1 \approx_E \varphi_2$ implique $([\![\varphi_1]\!]_{A_\eta}) = ([\![\varphi_2]\!]_{A_\eta})$;
- *inconditionnellement* $\not\vdash_E$-*sûre* ssi pour tout environnement φ et terme clos T tel que $\varphi \not\vdash_E T$, d'une part les tirages de φ et de T sont indépendants : pour tout ϕ_0, e_0, $\mathbb{P}[\phi_0, e_0 \xleftarrow{R} [\![\varphi, T]\!]_{A_\eta}] = \mathbb{P}[\phi_0 \xleftarrow{R} [\![\varphi]\!]_{A_\eta}] \times \mathbb{P}[e_0 \xleftarrow{R} [\![T]\!]_{A_\eta}]$, et d'autre part la distribution $(\xleftarrow{R} [\![T]\!]_{A_\eta})$ est sans collision.

D'une manière générale, la sûreté (inconditionnelle) vis-à-vis de $=_E$ est donnée par construction. En effet, les équations formelles vraies correspondent au comportement attendu des primitives cryptographiques, et sont donc valides dans le monde concret avec une probabilité 1. Le lemme suivant permet alors de conclure simplement.

Lemme 5.1 (Critère de $=_E$-sûreté)**.** *Soit E_0 un ensemble d'équations générant E et (A_η) une famille d'algèbres concrètes. On suppose que pour tout $l \doteq r$ dans E_0, on a* $\mathrm{names}(l, r) = \emptyset$ *et que pour tout η, pour tout environnement concret ϕ avec* $\mathrm{dom}(\phi) \supseteq \mathrm{var}(l, r)$, $[\![l]\!]_{\eta,\phi} = [\![r]\!]_{\eta,\phi}$.

Alors, (A_η) est inconditionnellement $=_E$-sûre.

Démonstration. Soit $\cong$ la relation entre termes définie par $T_1 \cong T_2$ ssi pour tout η, pour tout environnement concret ϕ,

$$\mathbb{P}\left[e_1, e_2 \leftarrow [\![T_1, T_2]\!]_{\eta,\phi} \ : \ e_1 = e_2\right] = 1.$$

Par hypothèse, $l \cong r$ pour tout $l \doteq r$ dans E_0. De plus, on vérifie facilement que $\cong$ est une congruence stable par substitution. Par conséquent, $\cong$ contient la relation $=_E$. □

En revanche, les autres propriétés sont généralement non triviales. Dans certains cas, il peut être nécessaire de restreindre les environnements ou les termes considérés par des contraintes de *bonne formation* pour pouvoir conclure à partir d'hypothèses cryptographiques standards : par exemple les travaux de M. Abadi et de P. Rogaway [AR00] excluent les cycles de chiffrements (voir la section 6.3).

5.2.2 Relations entre notions de sûreté et de fidélité

Nous étudions maintenant les relations entre les différentes notions définies plus haut.

Proposition 5.2. *Soit (A_η) une famille d'algèbres concrètes $=_E$-sûres. Alors*

1. *(A_η) est $\not=_E$-fidèle ;*
2. *si (A_η) est $=_E$-fidèle, alors (A_η) est $\approx_E$-fidèle.*

Démonstration.

1. Supposons que $\varphi \vdash_E T$, c'est-à-dire qu'il existe un terme M tel que $\mathrm{var}(M) \subseteq \mathrm{dom}(\varphi)$, $\mathrm{names}(M) \cap \mathrm{names}(\varphi, T) = \emptyset$ et $M\varphi =_E T$. On définit un adversaire $\mathcal{A}$ capable de déduire $[\![T]\!]$ à partir de $[\![\varphi]\!]$ de la manière suivante : étant donné un environnement concret $\phi = \{x_i \mapsto e_i\}$, $\mathcal{A}$ renvoie un échantillon $e \xleftarrow{R} [\![M]\!]_{A_\eta,\phi}$. Comme $(A_\eta)_{\eta\geq 0}$ est $=_E$-sûre, la probabilité de succès de $\mathcal{A}$ est écrasante.
2. Supposons que $\varphi_1 \not\approx_E \varphi_2$: il existe deux termes M et N tels que $\mathrm{var}(M, N) \subseteq \mathrm{dom}(\varphi_1)$, $\mathrm{names}(M, N) \cap \mathrm{names}(\varphi_1, \varphi_2) = \emptyset$ et par exemple $M\varphi_1 =_E N\varphi_1$ et $M\varphi_2 \neq_E N\varphi_2$. Soit $\mathcal{A}$ l'adversaire qui, étant donné η et ϕ, calcule $e \xleftarrow{R} [\![M]\!]_{A_\eta,\phi}$, $f \xleftarrow{R} [\![N]\!]_{A_\eta,\phi}$, et renvoie le résultat du test $e =_{A_\eta} f$. $\mathcal{A}$ fonctionne en temps polynomial ; de plus par $=_E$-sûreté et $=_E$-fidélité, son avantage est écrasant. □

Nous illustrons maintenant le fait que pour de nombreuses théories équationnelle, la sûreté vis-à-vis de $\approx_E$ implique toutes les autres notions de sûreté et de fidélité – ce qui au passage souligne l'importance de cette notion. Plus précisément, nous considérons une théorie arbitraire possédant des fonctions de hachage (indexées).

Proposition 5.3. *Soit (A_η) une famille $\approx_E$-sûre d'algèbres concrètes. Soit Key une sorte non dégénérée, et pour chaque sorte τ, $\mathsf{h}_\tau : \tau \times Key \to Hash$ un symbole libre. Supposons que le tirage aléatoire $(\xleftarrow{R} [\![Hash]\!]_{A_\eta})$ est sans collision. Alors*

1. *(A_η) est $=_E$-fidèle ;*

2. (A_η) *est* $\not\vdash_E$*-sûre ;*
3. *Supposons que l'implémentation des symboles* h_τ *soit résistante aux collisions (*collision-resistant*) au sens où pour tous* T_1, T_2 *de sorte* τ*, étant donné un nom frais* k *de sorte Key, la quantité*

$$\mathbb{P}\left[e_1, e_2, e_1', e_2' \xleftarrow{R} [\![T_1, T_2, \mathsf{h}_\tau(T_1, k), \mathsf{h}_\tau(T_2, k)]\!]_{A_\eta} \ : \ e_1 \neq_{A_\eta} e_2,\ e_1' =_{A_\eta} e_2'\right]$$

est négligeable. Alors (A_η) *est* $=_E$*-sûre,* $\not\vdash_E$*-fidèle et* $\approx_E$*-fidèle.*

Avant de prouver la proposition, nous commençons par établir plusieurs lemmes techniques sur l'équivalence statique.

Lemme 5.4. *Soit* T_1, T_2 *deux termes de sorte* τ *tels que* $T_1 \neq_E T_2$*. Soit* $\mathsf{h}_\tau : \tau \times Key \to Hash$ *un symbole libre tel que Key est non dégénérée. Considérons les environnements*

$$\begin{array}{rcl} \varphi_1 & = & \{x_1 \mapsto \mathsf{h}_s(T_1, k),\ x_2 \mapsto \mathsf{h}_s(T_2, k)\} \quad et \\ \varphi_2 & = & \{x_1 \mapsto n,\ x_2 \mapsto n'\} \end{array}$$

où k, n, n' *sont des noms frais distincts de sortes appropriées. On a l'équivalence :* $\varphi_1 \approx_E \varphi_2$*.*

Démonstration. Soit M et N deux termes tels que $\mathrm{var}(M, N) \subseteq \mathrm{dom}(\varphi)$ et $\mathrm{names}(M, N) \cap \mathrm{names}(\varphi_1, \varphi_2) = \emptyset$.

Supposons que $M\varphi_2 =_E N\varphi_2$. Soit $\theta = \{n \mapsto \mathsf{h}_s(T_1, k),\ n' \mapsto \mathsf{h}_s(T_2, k)\}$. Par le théorème 2.2 sur le remplacement des symboles libres, on a $M\varphi_1 = M(\varphi_2\theta) = (M\varphi_2)\theta =_E (N\varphi_2)\theta = N(\varphi_2\theta) = N\varphi_1$.

Réciproquement, supposons que $M\varphi_1 =_E N\varphi_1$. Soit ρ le remplacement modulo E : $\rho = \{\mathsf{h}_s(T_1, k) \mapsto_E n\}$. Par le théorème 2.2, on a $(M\varphi_1)\rho =_E (N\varphi_1)\rho$. Comme k n'apparaît pas dans M ni N, et que *Key* est non dégénérée, par le corollaire 2.4 appliqué à k, on obtient $(M\varphi_1)\rho = M(\varphi_1\rho)$ et $(N\varphi_1)\rho = N(\varphi_1\rho)$, d'où on déduit $M(\varphi_1\rho) =_E N(\varphi_1\rho)$.

Montrons maintenant que $\varphi_1\rho = \{x_1 = n,\ x_2 = \mathsf{h}_s(T_2, k)\}$. En effet, on a $(\mathsf{h}_s(T_2, k))\rho = \mathsf{h}_s(T_2\rho, k)$ puisque $T_1 \neq_E T_2$. De plus, comme k n'apparaît pas dans T_2, et que *Key* est non dégénérée, par le corollaire 2.4 appliqué à k, on a bien $T_2\rho = T_2$.

En appliquant $\rho' = \{\mathsf{h}_s(T_2, k) \mapsto_E n'\}$ à l'équation $M(\varphi_1\rho) =_E N(\varphi_1\rho)$, on obtient de la même manière d'une part $M(\varphi_1\rho\rho') =_E N(\varphi_1\rho\rho')$, et d'autre part $\varphi_1\rho\rho' = \varphi_2$. □

Lemme 5.5. *Soit* φ *un environnement et* T *un terme de sorte* τ*. Soit* $\mathsf{h}_\tau : \tau \times Key \to Hash$ *un symbole libre tel que Key est non dégénérée. Considérons les environnements*

$$\begin{array}{rcl} \varphi_1 & = & \varphi \cup \{x \mapsto \mathsf{h}_\tau(T, k), y \mapsto k\} \quad et \\ \varphi_2 & = & \varphi \cup \{x \mapsto n, y \mapsto k\} \end{array}$$

où x, y *sont des variables fraîches,* k *et* n *sont des noms frais de sortes appropriées. Si* $\varphi \not\vdash_E T$*, alors* $\varphi_1 \approx_E \varphi_2$*.*

Démonstration. Soit M et N deux termes tels que $\mathrm{var}(M,N) \subseteq \mathrm{dom}(\varphi)$ et $\mathrm{names}(M,N) \cap \mathrm{names}(\varphi_1,\varphi_2) = \emptyset$. Tout d'abord, comme pour le lemme 5.4, $M\varphi_2 =_E N\varphi_2$ implique $M\varphi_1 =_E N\varphi_1$.

Réciproquement, supposons que $M\varphi_1 =_E N\varphi_1$. Soit $\rho = \{\mathsf{h}_\tau(T,k) \mapsto_E n\}$. Par le théorème 2.2, on a $(M\varphi_1)\rho =_E (N\varphi_1)\rho$. Montrons que $(M\varphi_1)\rho = M(\varphi_1\rho)$.

En effet, dans le cas contraire, il existe un sous-terme M_1 de M tel que M_1 n'est pas une variable et $M_1\varphi_1 = \mathsf{h}_\tau(T',T'')$ avec $T' =_E T$ et $T'' =_E k$. Comme M_1 n'est pas une variable, M_1 est de la forme $M_1 = \mathsf{h}_\tau(M_1',M_1'')$ avec $M_1'\varphi_1 = T' =_E T$, ce qui implique que T est déductible ; contradiction.

On en déduit donc que $(M\varphi_1)\rho = M(\varphi_1\rho)$, et de manière identique $(N\varphi_1)\rho = N(\varphi_1\rho)$. En particulier, $M(\varphi_1\rho) =_E N(\varphi_1\rho)$. Comme *Key* est non dégénérée, et k n'apparaît pas dans φ, le corollaire 2.4 nous donne $\varphi\rho = \varphi$, ce qui implique $\varphi_1\rho = \varphi_2$ et donc la propriété voulue. □

Lemme 5.6. *Soit T_1, T_2 deux termes de sorte τ tels que $T_1 =_E T_2$. Soit $\mathsf{h}_\tau : \tau \times Key \to Hash$ un symbole libre. Soit*

$$\begin{aligned} \varphi_1 &= \{x_1 \mapsto \mathsf{h}_\tau(T_1,k),\ x_2 \mapsto \mathsf{h}_\tau(T_2,k)\} \\ \varphi_2 &= \{x_1 \mapsto n,\ x_2 \mapsto n\} \end{aligned}$$

où n est un nom frais de sorte Hash. Alors $\varphi_1 \approx_E \varphi_2$.

Démonstration. Soit M et N deux termes tels que $\mathrm{var}(M,N) \subseteq \mathrm{dom}(\varphi)$ et $\mathrm{names}(M,N) \cap (\mathrm{names}(\varphi) \cup \{n\}) = \emptyset$. Comme pour le lemme 5.4, $M\varphi_2 =_E N\varphi_2$ implique $M\varphi_1 =_E N\varphi_1$.

Réciproquement, supposons que $M\varphi_1 =_E N\varphi_1$. Soit $\rho = \{\mathsf{h}_\tau(T_1,k) \mapsto_E n\}$. Par le théorème 2.2, on a $(M\varphi_1)\rho =_E (N\varphi_1)\rho$. Comme k n'apparaît pas dans M ou N, par le corollaire 2.4, on a $(M\varphi_1)\rho = M(\varphi_1\rho)$ et $(N\varphi_1)\rho = N(\varphi_1)\rho$. Comme $T_1 =_E T_2$, on a $\varphi_1\rho = \varphi_2$, d'où $M\varphi_2 =_E N\varphi_2$. □

Nous pouvons maintenant établir la proposition 5.3 à l'aide des lemmes précédents.

Preuve de la proposition 5.3.

1. Soit T_1, T_2 deux termes de sorte s tels que $T_1 \neq_E T_2$. Considérons l'environnement $\varphi = \{x_1 \mapsto \mathsf{h}_\tau(T_1,k),\ x_2 \mapsto \mathsf{h}_\tau(T_2,k)\}$ où k est un nom frais de sorte *Key*. Comme $T_1 \neq_E T_2$ et h_τ est libre, nous avons $\varphi \approx_E \{x_1 \mapsto n,\ x_2 \mapsto n'\}$ où n, n' sont deux noms frais distincts de sorte *Hash* (lemme 5.4). Par hypothèse, ceci implique $[\![\varphi]\!] \approx [\![\{x_1 \mapsto n,\ x_2 \mapsto n'\}]\!]$. En particulier, on a

$$\begin{aligned} &\mathbb{P}\left[e_1, e_2 \xleftarrow{R} [\![T_1,\ T_2]\!]_{A_\eta} \ :\ e_1 =_{A_\eta} e_2 \right] \\ &\qquad \leq \mathbb{P}\left[e_1', e_2' \xleftarrow{R} [\![\mathsf{h}_\tau(T_1,k),\ \mathsf{h}_\tau(T_2,k)]\!]_{A_\eta} \ :\ e_1' =_{A_\eta} e_2' \right] \\ &\qquad\qquad \leq \mathbb{P}\left[e_1', e_2' \xleftarrow{R} [\![n,n']\!]_{A_\eta} \ :\ e_1' =_{A_\eta} e_2' \right] + f(\eta) \end{aligned}$$

pour une certaine fonction négligeable $f(\eta)$. Or, $(\xleftarrow{R} [\![\mathit{Hash}]\!]_{A_\eta})$ est sans collision. Les quantités ci-dessus sont donc négligeables.

2. Soit φ un environnement et T un terme clos de sorte τ tels que $\varphi \not\vdash_E T$. Soit $\varphi_0 = \varphi \cup \{x \mapsto \mathsf{h}_\tau(T,k), y \mapsto k\}$ et $\varphi_1 = \varphi \cup \{x \mapsto n, y \mapsto k\}$ où x, y sont deux variables fraîches, k est un nom frais de sorte *Key*, n est un nom frais de sort *Hash*. Comme $\varphi \not\vdash_E T$, on a $\varphi_0 \approx_E \varphi_1$ (lemme 5.5). Ainsi par hypothèse, $[\![\varphi_0]\!] \approx [\![\varphi_1]\!]$.
   Par contradiction, supposons qu'il existe un adversaire polynomial $\mathcal{A}$ capable de déduire $[\![T]\!]$ de $[\![\varphi]\!]$ avec une probabilité de succès non négligeable. Nous construisons un adversaire $\mathcal{B}$ pour distinguer $[\![\varphi_0]\!]$ et $[\![\varphi_1]\!]$ de la manière suivante : soit ϕ un échantillon de $[\![\varphi_b]\!]_\eta$ en entrée de $\mathcal{B}$ où $b \in \{0,1\}$. Soit $\widehat{T}$ la réponse de $\mathcal{A}$ étant donné la restriction de ϕ à $\mathrm{dom}(\varphi)$. $\mathcal{B}$ renvoie 0 si $x\phi =_{A_\eta} [\![\mathsf{h}_\tau]\!]_{A_\eta}(\widehat{T}, y\phi)$, et 1 sinon. Par définition, l'avantage de $\mathcal{B}$ est

$$\begin{aligned}
&\mathbb{P}[\phi \xleftarrow{R} [\![\varphi_0]\!]_\eta \ : \ \mathcal{B}(\eta,\phi) = 0] - \mathbb{P}[\phi \xleftarrow{R} [\![\varphi_1]\!]_\eta \ : \ \mathcal{B}(\eta,\phi) = 0] \\
&\quad = \quad \mathbb{P}[\phi \xleftarrow{R} [\![\varphi_0]\!]_\eta; \widehat{T} \xleftarrow{R} \mathcal{A}(\phi|_{\mathrm{dom}(\varphi)}) \ : \ x\phi =_{A_\eta} [\![\mathsf{h}_\tau]\!]_{A_\eta}(\widehat{T}, y\phi)] \\
&\qquad - \mathbb{P}[\phi \xleftarrow{R} [\![\varphi_1]\!]_\eta; \widehat{T} \xleftarrow{R} \mathcal{A}(\phi|_{\mathrm{dom}(\varphi)}) \ : \ x\phi =_{A_\eta} [\![\mathsf{h}_\tau]\!]_{A_\eta}(\widehat{T}, y\phi)] \\
&\quad \geq \quad \mathbb{P}[\phi, e \xleftarrow{R} [\![\varphi_0, T]\!]_\eta; \widehat{T} \xleftarrow{R} \mathcal{A}(\phi|_{\mathrm{dom}(\varphi)}) \ : \ \widehat{T} = e)] \\
&\qquad - \mathbb{P}[\phi \xleftarrow{R} [\![\varphi_1]\!]_\eta; \widehat{T} \xleftarrow{R} \mathcal{A}(\phi|_{\mathrm{dom}(\varphi)}) \ : \ x\phi =_{A_\eta} [\![\mathsf{h}_\tau]\!]_{A_\eta}(\widehat{T}, y\phi)]
\end{aligned}$$

   Notons que dans la dernière expression de probabilité $x\phi$ est tiré de la distribution $(\xleftarrow{R} [\![\mathit{Hash}]\!]_{A_\eta})$ indépendamment de $\widehat{T}$ et de $y\phi$. Par conséquent, comme la distribution $(\xleftarrow{R} [\![\mathit{Hash}]\!]_{A_\eta})$ est sans collision, l'avantage de $\mathcal{B}$ est non négligeable ; contradiction.
3. Soit T_1 et T_2 deux termes de sorte τ tels que $T_1 =_E T_2$. Considérons le même environnement que précédemment : $\varphi = \{x_1 \mapsto \mathsf{h}_\tau(T_1,k),\ x_2 \mapsto \mathsf{h}_\tau(T_2,k)\}$. Comme $T_1 =_E T_2$ et que h_τ est libre, on a $\varphi \approx_E \{x_1 \mapsto n,\ x_2 \mapsto n\}$ où n est un nom frais de sorte *Hash* (lemme 5.6). Par hypothèse cela implique que $[\![\varphi]\!] \approx [\![\{x_1 \mapsto n,\ x_2 \mapsto n\}]\!]$, donc

$$\mathbb{P}\left[e_1', e_2' \xleftarrow{R} [\![\mathsf{h}_\tau(T_1,k),\ \mathsf{h}_\tau(T_2,k)]\!]_{A_\eta} \ : \ e_1' =_{A_\eta} e_2' \right] \geq 1 - f(\eta)$$

   où $f(\eta)$ est négligeable. Comme l'implémentation de h_τ est résistante aux collision, on en déduit que

$$\mathbb{P}\left[e_1, e_2 \xleftarrow{R} [\![T_1,\ T_2]\!]_{A_\eta} \ : \ e_1 \neq_{A_\eta} e_2 \right]$$

   est négligeable, donc (A_η) est $=_E$-sûre. Les autres propriétés découlent de la proposition 5.2. □

5.3 Implications de la sûreté vis-à-vis de l'équivalence statique dans les groupes abéliens

Dans cette section, nous étudions certaines conséquences de la sûreté vis-à-vis de l'équivalence statique dans les groupes abéliens. Notamment, par analogie avec les

travaux de Hohenberger et Rivest sur la pseudo-liberté (*pseudo-freeness*) [Hoh03, Riv04], nous montrons que la $\approx_E$-sûreté implique plusieurs hypothèses cryptographiques standards dans les groupes abéliens.

En guise de modèle abstrait, nous considérons le groupe abélien libre G dont les exposants sont munis d'une structure d'anneau A (libre, compatible avec les lois de groupe). Formellement, ceci correspond à la signature suivante :

$$\begin{array}{lcllcl} * & : & G \times G \to G & - & : & A \to A \\ 1_G & : & G & \cdot & : & A \times A \to A \\ + & : & A \times A \to A & 1_A & : & A \\ 0_A & : & A & \mathsf{exp} & : & G \times A \to G \end{array}$$

à laquelle on ajoute un nombre infini de noms de sortes G et A, notés respectivement $g, g_1 \ldots$ et $a, b, a_1 \ldots$

Nous utilisons la notation infixe pour les opérateurs $*$, $\cdot$, $+$, et écrivons g^a au lieu de $\mathsf{exp}(g, a)$. Notez que l'opération d'inversion dans G est représentable par $x \mapsto \mathsf{exp}(x, -(1_A)) = x^{-(1_A)}$. Les termes construits sur cette signature sont munis de la théorie équationnelle E_{G} engendrée par les équations suivantes (où x, y, z sont des variables de sorte G, et u, v, w des variables de sorte A) :

$$\begin{array}{rcl@{\qquad}rcl} u + v & \doteq & v + u & x * y & \doteq & y * x \\ u + (v + w) & \doteq & (u + v) + w & x * (y * z) & \doteq & (x * y) * z \\ u + 0_A & \doteq & u & x * 1_G & \doteq & x \\ u + (-u) & \doteq & 0_A & (x^u)^v & \doteq & x^{(u \cdot v)} \\ u \cdot v & \doteq & v \cdot u & x^u * x^v & \doteq & x^{u+v} \\ u \cdot (v \cdot w) & \doteq & (u \cdot v) \cdot w & x^{1_A} & \doteq & x \\ u \cdot 1_A & \doteq & u & x^{0_A} & \doteq & 1_G \\ (u + v) \cdot w & \doteq & u \cdot w + v \cdot w & (x * y)^u & \doteq & x^u * y^u \end{array}$$

Nous rappelons maintenant (brièvement) plusieurs problèmes classiques sur les groupes. Du point de vue des applications cryptographiques, il est généralement désirable que ces problèmes soient *difficiles*, c'est-à-dire irréalisable par un adversaire en temps polynomial probabiliste (sauf pour une proportion négligeable de cas[3]) :

- le problème du *logarithme discret* (DL) : étant donnés g et g', trouver a, tel que $g^a = g'$;
- le problème *Diffie-Hellman calculatoire* (CDH) : étant donnés g, g^a et g^b, trouver g^{ab} ;
- le problème *Diffie-Hellman décisionnel* (DDH) : étant donnés g, g^a et g^b, distinguer g^{ab} d'un élément aléatoire g^c avec une probabilité significativement plus grande que $\frac{1}{2}$;
- le problème *Rivest-Shamir-Adleman* (RSA) : étant donnés a et g^a, trouver g.

Proposition 5.7. *Soit (A_η) une famille d'algèbres concrètes sur la signature ci-dessus qui soit $\approx_{E_{\mathsf{G}}}$-sûre. Alors les problèmes DL, CDH et DDH sont difficiles dans (A_η).*

Démonstration. En effet, considérons les deux environnements

$$\varphi_1 = \{x_1 \mapsto g, x_2 \mapsto g^a, x_3 \mapsto g^b, x_4 \mapsto g^{a \cdot b}\} \text{ et}$$
$$\varphi_2 = \{x_1 \mapsto g, x_2 \mapsto g^a, x_3 \mapsto g^b, x_4 \mapsto g^c\}.$$

[3] Nous nous référons à la variante de ces problèmes où les valeurs concrètes g, g', a, b ci-dessous sont tirées au sort.

où a, b, c sont des noms de sorte A et g est un nom de sorte G.

Le problème de distinguer les sémantiques concrètes de ces deux environnements encode exactement le problème DDH. Pour des raisons de lisibilité, nous prouvons séparément l'équivalence statique $\varphi_1 \approx_{E_\mathsf{G}} \varphi_2$ (lemme 5.10). Par $\approx_{E_\mathsf{G}}$-sûreté, on a donc $(\llbracket\varphi_1\rrbracket_{A_\eta}) \approx (\llbracket\varphi_2\rrbracket_{A_\eta})$; autrement dit le problème DDH est difficile.

Par ailleurs, il est bien connu que la difficulté de DDH implique celle des problèmes CDH et DL. □

Le cas de RSA est traité de manière identique, pourvu que l'on augmente la signature d'un symbole libre modélisant une fonction à sens unique.

Proposition 5.8. *Soit (A_η) une famille $\approx_{E_\mathsf{G}}$-sûre d'algèbres concrètes sur la signature précédente augmentée d'un symbole libre $\mathsf{h} : G \to Hash$. Alors, le problème RSA est difficile dans (A_η).*

Démonstration. Considérons les deux environnements

$$\varphi_1 = \{x_1 \mapsto g^a, x_2 \mapsto a, x_3 \mapsto \mathsf{h}(g)\} \text{ et}$$
$$\varphi_2 = \{x_1 \mapsto g^a, x_2 \mapsto a, x_3 \mapsto h\}.$$

où g, a, h sont des noms de sortes appropriées. Nous prouvons plus bas que $\varphi_1 \approx_{E_\mathsf{G}} \varphi_2$ (lemme 5.11). Par $\approx_{E_\mathsf{G}}$-sûreté, on a donc $(\llbracket\varphi_1\rrbracket_{A_\eta}) \approx (\llbracket\varphi_2\rrbracket_{A_\eta})$. Il s'ensuit que le problème RSA est difficile dans (A_η). (En effet, une fois g calculé, il est évident de distinguer $\mathsf{h}(g)$ d'un nombre aléatoire h.) □

Une question ouverte est de comparer la $\approx_{E_\mathsf{G}}$-sûreté avec la notion de groupes pseudo-libres de R. Rivest [Riv04]. Nous conjecturons que ces deux notions sont en réalité incomparables. En effet, d'une part notre notion implique (facilement) la difficulté de DDH, alors que cette question reste ouverte pour les groupes pseudo-libres. D'autre part, les groupes pseudo-libres mettent en jeu un attaquant adaptatif (dans une certaine mesure), alors que l'adversaire de notre modèle est purement passif.

Récemment, D. Micciancio [Mic05] a montré sous l'hypothèse RSA *forte*, que $(\mathbb{Z}/N)^*$ était pseudo-libre lorsque N était un entier RSA (plus exactement un produit de deux nombres premiers forts). L'autre implication étant établie dans le papier originel de Rivest [Riv04], dans ces conditions, l'hypothèse RSA forte est donc équivalente à l'hypothèse de pseudo-liberté. Pour autant, la $\approx_{E_\mathsf{G}}$-sûreté (même dans la signature augmentée un symbole libre) n'implique pas nécessairement la pseudo-liberté, car la proposition 5.8 montre la difficulté du problème RSA (simple).

5.3.1 Preuves d'équivalence statique dans les groupes

Nous établissons maintenant les propriétés d'équivalence statique utilisées plus haut dans la théorie équationnelle E_G. Pour cela, nous commençons par caractériser les classes d'équivalence de E_G à l'aide d'un lemme de représentation.

Soit $\mathcal{X}_A$ (respectivement $\mathcal{X}_G$ et $\mathcal{X}_{Hash}$) l'ensemble des variables de sortes A (respectivement G et *Hash*). Soit $\mathcal{N}_A$ (respectivement $\mathcal{N}_G$ et $\mathcal{N}_{Hash}$) l'ensemble des noms de sortes A (respectivement G et *Hash*). On appelle AC la théorie équationnelle engendrée par les équations d'associativité et de commutativité des symboles $+$, $\cdot$, $\times$.

On appelle *monôme unitaire de sorte A* une fonction $\beta : \mathcal{X}_A \cup \mathcal{N}_A \to \mathbb{N}$ nulle presque partout (c'est-à-dire nulle sauf pour un nombre fini d'entrées). Une telle fonction β est vue comme un terme de sorte A (modulo AC) :

$$\beta =_{AC} \prod_{\substack{a \in \mathcal{N}_A \\ \beta(a) \neq 0}} a^{\beta(a)} \cdot \prod_{\substack{u \in \mathcal{X}_A \\ \beta(u) \neq 0}} u^{\beta(u)}$$

où les produits vides sont vues comme le terme 1_A, et la notation $a^{\beta(a)}$ $(\beta(a) \neq 0)$ représente le terme $\underbrace{a \cdot \cdots \cdot a}_{\beta(a) \text{ fois}}$ (bien défini modulo AC). L'ensemble des monômes unitaires de sorte A est noté $\mathcal{M}_A$.

Une *forme canonique de sorte A* une fonction $\alpha : \mathcal{M}_A \to \mathbb{Z}$ nulle presque partout. Un tel α est vue comme un terme de sorte A (modulo AC) :

$$\alpha =_{AC} \sum_{\substack{\beta \in \mathcal{M}_A \\ \alpha(\beta) \neq 0}} \alpha(\beta) \cdot \beta$$

où la somme vide représente le terme 0_A, et les entiers sont vus de manière évidente comme des termes clos 0_A, $1_A + \cdots + 1_A$ ou $-(1_A + \cdots + 1_A)$ de sorte A.

Une *forme canonique de sorte G* est une fonction γ allant de $\mathcal{X}_N \cup \mathcal{N}_N$ vers les formes canoniques de sorte A, nulle presque partout (c'est-à-dire valant la fonction constante 0 sauf pour un nombre fini d'entrées). Une forme canonique γ est vue comme un terme de sorte G (modulo AC) :

$$\gamma =_{AC} \prod_{\substack{g \in \mathcal{N}_G \\ \gamma(g) \neq 0}} g^{\gamma(g)} * \prod_{\substack{x \in \mathcal{X}_G \\ \gamma(x) \neq 0}} x^{\gamma(x)}$$

où les produits vides sont vues comme le terme 1_G.

Une *forme canonique de sorte Hash*, notée ι, est une variable de sorte *Hash* : $\iota = z \in \mathcal{X}_{Hash}$, un nom de sorte *Hash* : $\iota = h \in \mathcal{N}_{Hash}$, ou bien une forme canonique γ de sorte G vue comme le terme $\iota = \mathsf{h}(\gamma)$.

Lemme 5.9. *Pour tout terme T de sorte A (respectivement G, Hash), il existe une unique forme canonique α_T (respectivement γ_T, ι_T) telle que*

$$T =_{E_\mathsf{G}} \alpha_T$$

(respectivement $T =_{E_\mathsf{G}} \gamma_T$, $T =_{E_\mathsf{G}} \iota_T$).

Schéma de preuve. On montre l'existence d'une forme canonique de T par récurrence sur la structure de T. Par exemple, étant donné $T = T_1 * T_2$, et deux formes canoniques α_{T_1} et α_{T_2}, on obtient une forme canonique de T en réarrangeant le produit $\alpha_{T_1} * \alpha_{T_2}$ modulo E_G (en utilisant au besoin l'hypothèse d'induction sur les exposants). Concernant l'unicité, il suffit de montrer que si deux formes canoniques sont égales en tant que termes modulo E_G, alors elles sont égales (en tant que fonctions). Formellement, on établit ce fait en étudiant la forme AC-normale associée à chaque forme canonique par le système de réécriture AC-convergent suivant :

$$\begin{array}{rcl@{\qquad}rcl}
u + 0_A & \to & u & x * 1_G & \to & x \\
u + (-u) & \to & 0_A & (x^u)^v & \to & x^{(u \cdot v)} \\
u \cdot 1_A & \to & u & x^u * x^v & \to & x^{u+v} \\
(u+v) \cdot w & \to & u \cdot w + v \cdot w & x^{1_A} & \to & x \\
u \cdot 0_A & \to & 0_A & x^{0_A} & \to & 1_G \\
-(u+v) & \to & (-u) + (-v) & (x * y)^u & \to & x^u * y^u \\
(-u) \cdot v & \to & -(u \cdot v) & x * x & \to & x^{(1_A + 1_A)} \\
-(-u) & \to & u & x * x^u & \to & x^{u+1_A} \\
-0_A & \to & 0_A & (1_G)^u & \to & 1_G
\end{array}$$

Ce système a été obtenu en orientant et en complétant les équations générant E_{G} autres que AC à l'aide de l'outil Cime [CMMU00]. □

Proposition 5.10. *Soit* $\varphi_1 = \{x_1 \mapsto g, x_2 \mapsto g^a, x_3 \mapsto g^b, x_4 \mapsto g^{a \cdot b}\}$ *et* $\varphi_2 = \{x_1 \mapsto g, x_2 \mapsto g^a, x_3 \mapsto g^b, x_4 \mapsto g^c\}$ *où* $g, a, b, c \in \mathcal{N}$. *On a* $\varphi_1 \approx_{E_{\mathsf{G}}} \varphi_2$.

Démonstration. Soit M, N deux termes de même sorte tels que $\mathrm{var}(M, N) \subseteq \mathrm{dom}(\varphi_1)$ et $\mathrm{names}(M, N) \cap \mathrm{names}(\varphi_1, \varphi_2) = \emptyset$.

Supposons $M\varphi_2 =_{E_{\mathsf{G}}} N\varphi_2$. Par la proposition 2.5, en considérant le remplacement $\{c \mapsto a \cdot b\}$ (sachant que $c \notin \mathrm{names}(M, N)$), on obtient $M\varphi_1 =_{E_{\mathsf{G}}} N\varphi_1$.

Réciproquement, supposons $M\varphi_1 =_{E_{\mathsf{G}}} N\varphi_1$. Si M et N sont de sorte A, alors $\mathrm{var}(M, N) = \emptyset$ et donc a fortiori $M\varphi_2 = M\varphi_1 =_E N\varphi_1 = N\varphi_2$.

Sinon, M et N sont de sorte G. Sachant que $M\varphi_1 =_{E_{\mathsf{G}}} N\varphi_1$ équivaut à $M\varphi_1 * (N\varphi_1)^{-1_A} = 1_G$, on peut supposer que $N = 1_G$.

Sachant que $\mathrm{var}(M) \subseteq \mathrm{dom}(\varphi_1)$ et $\mathrm{names}(M) \cap \mathrm{names}(\varphi_1, \varphi_2) = \emptyset$, la forme canonique γ de M s'écrit :

$$M =_{E_{\mathsf{G}}} \prod_{g' \neq g} {g'}^{\gamma(g')} * x_1^{\gamma(x_1)} * \ldots * x_4^{\gamma(x_4)}$$

où les $\gamma(g')$ et $\gamma(x_i)$ représentent des termes clos dont les noms sont disjoints de $\{a, b, c\}$. On a donc

$$M\varphi_1 =_{E_{\mathsf{G}}} \prod_{g' \neq g} {g'}^{\gamma(g')} * g^{\gamma(x_1) + \gamma(x_2) \cdot a + \gamma(x_3) \cdot b + \gamma(x_4) a \cdot b} =_{E_{\mathsf{G}}} 1_G$$

d'où on déduit pour tout i, $\gamma(x_i) = 0_A$ et pour tout g', $\gamma(g') = 0_A$, c'est-à-dire $M = 1_G$. □

Proposition 5.11. *On suppose ici la signature et la théorie équationnelle* E_{G} *augmentées d'un symbole libre unaire* h. *Soit* $\varphi_1 = \{x_1 \mapsto g^a, x_2 \mapsto a, x_3 \mapsto \mathsf{h}(g)\}$ *et* $\varphi_2 = \{x_1 \mapsto g^a, x_2 \mapsto a, x_3 \mapsto h\}$ *où* $g, a, h \in \mathcal{N}$. *On a* $\varphi_1 \approx_{E_{\mathsf{G}}} \varphi_2$.

Démonstration. Soit M, N deux termes de même sorte tels que $\mathrm{var}(M, N) \subseteq \mathrm{dom}(\varphi_1)$ et $\mathrm{names}(M, N) \cap \mathrm{names}(\varphi_1, \varphi_2) = \emptyset$.

Supposons $M\varphi_2 =_{E_{\mathsf{G}}} N\varphi_2$. Par la proposition 2.5, en considérant le remplacement $\{h \mapsto \mathsf{h}(g)\}$ (sachant que $h \notin \mathrm{names}(M, N)$), on obtient $M\varphi_1 =_{E_{\mathsf{G}}} N\varphi_1$.

Réciproquement, supposons $M\varphi_1 =_{E_{\mathsf{G}}} N\varphi_1$. Si M et N sont de sorte A ou G, alors $\mathrm{var}(M, N) \subseteq \{x_1, x_2\}$ et donc $M\varphi_2 = M\varphi_1 =_E N\varphi_1 = N\varphi_2$.

Sinon, M et N sont de sorte *Hash*. On peut supposer $M = x_3$ et $N = \mathsf{h}(N')$ où $\mathrm{var}(N') \subseteq \{x_1, x_2\}$ (les autres cas sont évidents). Sachant que h est libre, par le corollaire 2.3, $M\varphi_1 =_{E_\mathsf{G}} N\varphi_1$ équivaut à $N'\varphi_1 =_{E_\mathsf{G}} g$.

Sachant que $\mathrm{var}(N') \subseteq \{x_1, x_2\}$ et $\mathrm{names}(N') \cap \mathrm{names}(\varphi_1, \varphi_2) = \emptyset$, la forme canonique γ de N' est de la forme :

$$N' =_{E_\mathsf{G}} \prod_{g' \neq g} {g'}^{\gamma(g')} * x_1^{\gamma(x_1)}$$

où les $\gamma(g')$ et $\gamma(x_1)$ représentent des termes ayant x_2 comme seule variable possible, et ne contenant pas a. On a donc

$$N'\varphi_1 =_{E_\mathsf{G}} \prod_{g' \neq g} {g'}^{\gamma(g')\{x_2 \mapsto a\}} * g^{\gamma(x_1)\cdot a}$$

ce qui contredit le fait que $N'\varphi_1 =_{E_\mathsf{G}} g$. □

5.4 Étude d'un critère de sûreté pour l'équivalence statique

Nous présentons maintenant plusieurs techniques pour établir des propriétés de sûreté vis-à-vis de l'équivalence statique. Notamment, nous présentons un critère suffisant dans la section 5.4.1 et établissons un résultat de complétude partielle de ce critère dans la section 5.4.2.

5.4.1 Sémantique idéale et critère de $\approx_E$-sûreté

Étant donné une implémentation pour chaque primitive, nous avons défini dans la section 5.1.3 la sémantique concrète $[\![\varphi]\!]_{A_\eta}$ des environnements φ. Nous leur donnons maintenant une seconde sémantique, dite *idéale*, définie intuitivement comme le tirage aléatoire d'une valeur concrète (dans un espace adéquat) conditionné par l'ensemble des tests formels vrais dans φ.

Plus précisément, pour chaque environnement φ, nous notons $\mathrm{eq}_E(\varphi)$ l'ensemble des *tests valides* de φ :

$$\mathrm{eq}_E(\varphi) \triangleq \{M \doteq N \in \mathrm{test}(\varphi) \mid M\varphi =_E N\varphi\}$$

où $\mathrm{test}(\varphi) \triangleq \{M \doteq N \mid \mathrm{var}(M,N) \subseteq \mathrm{dom}(\varphi),\ \mathrm{names}(M,N) \cap \mathrm{names}(\varphi) = \emptyset\}$. Notez que, par définition, $\varphi \approx_E \varphi'$ ssi $\mathrm{eq}_E(\varphi) \cap \mathrm{test}(\varphi') = \mathrm{eq}_E(\varphi') \cap \mathrm{test}(\varphi)$.

Soit (A_η) une famille d'algèbres concrètes, $\varphi = \{x_1 \mapsto T_1, \ldots, x_n \mapsto T_n\}$ un environnement, et $\tau_i = \mathrm{sort}(x_i)$. On définit l'ensemble des *valeurs concrètes* de φ par

$$\mathrm{Val}_{A_\eta}(\varphi) \triangleq \left\{\{x_1 \mapsto e_1, \ldots, x_n \mapsto e_n\} \mid (e_1, \ldots, e_n) \in [\![\tau_1]\!]_{A_\eta} \times \cdots \times [\![\tau_n]\!]_{A_\eta}\right\}$$

et l'on note $\phi \xleftarrow{R} [\![\varphi]\!]^{\mathrm{val}}_{A_\eta}$ le tirage d'une valeur aléatoire $\phi = \{x_1 \mapsto e_1, \ldots, x_n \mapsto e_n\}$ dans $\mathrm{Val}_{A_\eta}(\varphi)$ en utilisant les tirages $e_i \xleftarrow{R} [\![\tau_i]\!]_{A_\eta}$ de la manière naturelle.

Notons $\mathrm{Val}'_{A_\eta}(\varphi)$ le sous-ensemble des valeurs concrètes de φ, qui satisfont tous les tests valides de φ :

$$\mathrm{Val}'_{A_\eta}(\varphi) \triangleq \Big\{\phi \in \mathrm{Val}_{A_\eta}(\varphi) \mid \forall (M,N) \in \mathrm{eq}_E(\varphi), \\ \mathbb{P}\left[e_1, e_2 \xleftarrow{R} [\![M,N]\!]_{A_\eta,\{x_1\mapsto e_1,\ldots,x_n\mapsto e_n\}} \;:\; e_1 = e_2\right] = 1\Big\}$$

Notons que si (A_η) est inconditionnellement $=_E$-sûre, $\mathrm{Val}'_{A_\eta}(\varphi)$ est non vide puisqu'il contient au moins les valeurs de la sémantique concrète (normale) de φ.

Définition 5.5 (Sémantique idéale)**.** Soit (A_η) une famille inconditionnellement $=_E$-sûre d'algèbres concrètes et φ un environnement. La *sémantique idéale* de φ est la famille de distributions $([\![\varphi]\!]^{ideal}_{A_\eta})$, où pour chaque η, la probabilité de choisir $\phi \in \mathrm{Val}_{A_\eta}(\varphi)$ est définie par

$$\mathbb{P}[\phi \leftarrow [\![\varphi]\!]^{ideal}_{A_\eta}] \triangleq \begin{cases} 0 & \text{si } \phi \notin \mathrm{Val}'_{A_\eta}(\varphi) \\ \dfrac{\mathbb{P}[\phi \xleftarrow{R} [\![\varphi]\!]^{\mathrm{val}}_{A_\eta}]}{\sum_{\phi_0 \in \mathrm{Val}'(\varphi)} \mathbb{P}[\phi_0 \xleftarrow{R} [\![\varphi]\!]^{\mathrm{val}}_{A_\eta}]} & \text{sinon} \end{cases}$$

Nous commençons par décrire une caractérisation simple de la sémantique idéale dans le cas des distributions uniformes.

On dit que (A_η) *est à distributions uniformes* ssi pour tout η et toute sorte τ, $[\![\tau]\!]_{A_\eta}$ est un ensemble fini, $=_{A_\eta,\tau}$ est la relation usuelle d'égalité, et la distribution associée à τ par A_η est la distribution uniforme sur $[\![\tau]\!]_{A_\eta}$.

Proposition 5.12. *Soit (A_η) une famille inconditionnellement $=_E$-sûre d'algèbres concrètes à distributions uniformes. Soit φ un environnement. La* sémantique idéale $[\![\varphi]\!]^{ideal}_{A_\eta}$ *de φ est la distribution uniforme sur l'ensemble (fini, non vide) des valeurs concrètes de φ,* $\mathrm{Val}'_{A_\eta}(\varphi)$.

Par exemple, soit $\varphi = \{x_1 \mapsto n_1, x_2 \mapsto n_2\}$ où n_1 et n_2 sont des noms de sorte τ. La théorie E étant stable par remplacement des noms (proposition 2.5), on a $\mathrm{eq}_E(\varphi) = \{M \doteq N \in \mathrm{test}(\varphi) \mid M =_E N\}$. Comme (A_η) est inconditionnellement $=_E$-sûre, on en déduit que $[\![\varphi]\!]^{ideal}_{A_\eta}$ est simplement la distribution uniforme sur $[\![\tau]\!]_{A_\eta} \times [\![\tau]\!]_{A_\eta}$.

Preuve de la proposition 5.12. Comme (A_η) est à distributions uniformes, on a $\mathbb{P}[\phi \xleftarrow{R} [\![\varphi]\!]^{\mathrm{val}}_{A_\eta}] = \frac{1}{|\mathrm{Val}_{A_\eta}(\varphi)|}$. Ainsi, la probabilité de tirer $\phi \in \mathrm{Val}'_{A_\eta}(\varphi)$ s'écrit

$$\mathbb{P}[\phi \leftarrow [\![\varphi]\!]^{ideal}_{A_\eta}] = \frac{\frac{1}{|\mathrm{Val}_{A_\eta}(\varphi)|}}{\sum_{\phi \in \mathrm{Val}'_{A_\eta}(\varphi)} \frac{1}{|\mathrm{Val}_{A_\eta}(\varphi)|}} = \frac{1}{|\mathrm{Val}'_{A_\eta}(\varphi)|}$$

ce qui correspond à la distribution uniforme sur $\mathrm{Val}'_{A_\eta}(\varphi)$. □

Nous pouvons maintenant énoncer un critère suffisant de $\approx_E$-sûreté : intuitivement, les deux sémantiques, concrète et idéale, doivent être indistinguables.

Théorème 5.13 (Critère de $\approx_E$-sûreté)**.** *Soit (A_η) une famille inconditionnellement $=_E$-sûre d'algèbres concrètes. Supposons que tout environnement φ satisfait $([\![\varphi]\!]_{A_\eta}) \approx ([\![\varphi]\!]^{ideal}_{A_\eta})$. Alors (A_η) est $\approx_E$-sûr.*

Démonstration. Soit $\varphi_1 \approx_E \varphi_2$. L'égalité $\mathrm{eq}_E(\varphi_1) \cap \mathrm{test}(\varphi_2) = \mathrm{eq}_E(\varphi_2) \cap \mathrm{test}(\varphi_1)$ implique pour tout η, $\mathrm{Val}'_{A_\eta}(\varphi_1) = \mathrm{Val}'_{A_\eta}(\varphi_2)$; ainsi les distributions $[\![\varphi_1]\!]^{ideal}_{A_\eta}$ et $[\![\varphi_2]\!]^{ideal}_{A_\eta}$ sont égales. Nous utilisons la transitivité de la relation d'indistinguabilité $\approx$ pour conclure : $([\![\varphi_1]\!]_{A_\eta}) \approx ([\![\varphi_1]\!]^{ideal}_{A_\eta}) = ([\![\varphi_2]\!]^{ideal}_{A_\eta}) \approx ([\![\varphi_2]\!]_{A_\eta})$. □

5.4.2 Environnements transparents

Dans cette section nous montrons la complétude du critère précédent pour une classe de théories équationnelles appelées théories transparentes. Pour cela, nous introduisons la notion générale d'*environnements transparents*. Cette notion s'avèrera également utile pour les applications du chapitre 6.

Définition 5.6 (Environnement transparent)**.** Un environnement φ est *transparent* pour une théorie équationnelle E ssi tout sous-terme de φ est déductible de φ dans E.

Exemple 5.1. Dans la théorie E_{enc}, étant donné $k_1, k_2, k_3, k_4 \in \mathcal{N}$, l'environnement

$$\varphi_1 = \{x_1 \mapsto \mathsf{enc}(\mathsf{enc}(k_4, k_3), k_1),\ x_2 \mapsto \mathsf{enc}(k_1, k_2),\ x_3 \mapsto k_2\}$$

n'est pas transparent, car k_3 et k_4 ne sont pas déductibles ; en revanche l'environnement

$$\overline{\varphi_1} = \{x_1 \mapsto \mathsf{enc}(n_1, k_1), x_2 \mapsto \mathsf{enc}(k_1, k_2), x_3 \mapsto k_2\}$$

est transparent. Remarquons au passage que $\varphi_1 \approx_{E_{\mathsf{enc}}} \overline{\varphi_1}$.

La proposition suivante permet de caractériser finiment l'ensemble des tests valides d'un environnement transparent. Rappelons que nous considérons une signature dans laquelle les seuls symboles de fonction non publics sont les noms.

Proposition 5.14. *Soit φ un environnement transparent dans E. Alors, φ est de la forme*

$$\varphi = \{x_1 \mapsto C_1[a_1, \ldots, a_m], \ldots, x_n \mapsto C_n[a_1, \ldots, a_m]\}$$

où $C_1, \ldots, C_n$ sont des contextes publics et les $a_1, \ldots, a_m$ sont des noms distincts et déductibles de φ : $\varphi \vdash_E a_i$.

Pour chaque a_i, soit M_{a_i} un terme tel que
- $\mathrm{var}(M_{a_i}) \subseteq \{x_1, \ldots, x_n\}$,
- $\mathrm{names}(M_{a_i}) \cap \{a_1, \ldots, a_m\} = \emptyset$, *et*
- $M_{a_i}\varphi =_E a_i$.

Alors, tout test valide de φ est une conséquence logique de la théorie E et des tests $x_j \doteq C_j[M_{a_1}, \ldots, M_{a_m}]$ $(1 \leq j \leq n)$, au sens de la logique équationnelle (table 2.1) – les x_j étant considérés ici comme des constantes :

$$E \cup \{x_j \doteq C_j[M_{a_1}, \ldots, M_{a_m}] \mid 1 \leq j \leq n\} \vdash \mathrm{eq}_E(\varphi).$$

Démonstration. Soit $M \doteq N \in \mathrm{eq}_E(\varphi)$. Par définition, on a $M\varphi =_E N\varphi$, c'est-à-dire

$$M\{x_j \mapsto C_j[a_1, \ldots, a_m]\}_{1 \leq j \leq n} =_E N\{x_j \mapsto C_j[a_1, \ldots, a_m]\}_{1 \leq j \leq n}$$

Comme E est stable par remplacement des noms (proposition 2.5), on a en particulier

$$M\{x_j \mapsto C_j[M_{a_1}, \ldots, M_{a_m}]\}_{1\leq j\leq n} =_E N\{x_j \mapsto C_j[M_{a_1}, \ldots, M_{a_m}]\}_{1\leq j\leq n}.$$

D'où on déduit par transitivité,

$$E \cup \{x_j \doteq C_j[M_{a_1}, \ldots, M_{a_m}] \mid 1 \leq j \leq n\} \vdash M \doteq N \qquad \square$$

Une autre propriété essentielle des environnements transparents est que leurs sémantiques concrète et idéale coïncident systématiquement dans le cas des distributions uniformes.

Proposition 5.15. *Soit (A_η) une famille inconditionnellement $=_E$-sûre d'algèbres concrètes, à distributions uniformes. Soit φ un environnement transparent. Les sémantiques concrète et idéale de φ engendrent la même famille de distributions : pour tout η, $[\![\varphi]\!]_{A_\eta} = [\![\varphi]\!]^{ideal}_{A_\eta}$.*

Démonstration. Soit $\varphi = \{x_1 \mapsto C_1[a_1, \ldots, a_m], \ldots, x_n \mapsto C_n[a_1, \ldots, a_m]\}$, où $M_i\varphi =_E a_i$ $(1 \leq i \leq m)$ comme précédemment. Soit $\tau_i = \mathrm{sort}(a_i)$, $\tau'_j = \mathrm{sort}(x_j)$ et η le paramètre de complexité.

La sémantique concrète (normale) de φ consiste à associer une valeur dans $\mathcal{F} = \mathrm{Val}_{A_\eta}(\varphi)$ à tout tirage aléatoire des valeurs des noms, pris dans $\mathcal{E} = [\![\tau_1]\!]_{A_\eta} \times \cdots \times [\![\tau_m]\!]_{A_\eta}$. Notons $\alpha : \mathcal{E} \to \mathcal{F}$ cette fonction, définie par

$$\alpha(e_1, \ldots, e_m) = \Big\{x_1 \mapsto [\![C_1[y_1, \ldots, y_m]]\!]_{\{y_1\mapsto e_1, \ldots, y_m\mapsto e_m\}}, \ldots, \\ x_n \mapsto [\![C_n[y_1, \ldots, y_m]]\!]_{\{y_1\mapsto e_1, \ldots, y_m\mapsto e_m\}}\Big\}$$

où les y_i sont des variables fraîches de sorte τ_i (respectivement), et où nous omettons l'indice A_η par souci de lisibilité.

À l'aide des M_i, nous pouvons aussi définir une fonction $\beta : \mathcal{F} \to \mathcal{E}$:

$$\beta(\phi) = \Big([\![M_1]\!]_\phi, \ldots, [\![M_m]\!]_\phi\Big)$$

Comme $M_i\varphi =_E a_i$, que (A_η) sont inconditionnellement $=_E$-sûre, et que la distribution sur $\mathcal{E}$ n'a aucun élément de poids nul, on a par construction $\beta \circ \alpha = Id_{\mathcal{E}}$. Par conséquent α est injective et induit une bijection de $\mathcal{E}$ vers son image $\mathcal{G} = \alpha(\mathcal{E})$. De plus, $\mathcal{G}$ satisfait

$$\begin{aligned}
\mathcal{G} &= \{\phi \in \mathcal{F} \mid \alpha(\beta(\phi)) = \phi\} \\
&= \Big\{\phi \in \mathcal{F} \mid \forall j, [\![C_j[y_1, \ldots, y_m]]\!]_{\{y_1=[\![M_1]\!]_\phi, \ldots, y_m=[\![M_i]\!]_\phi\}} = [\![x_j]\!]_\phi\Big\} \\
&= \Big\{\phi \in \mathcal{F} \mid \forall j, [\![C_j[M_1, \ldots, M_m]]\!]_\phi = [\![x_j]\!]_\phi\Big\}
\end{aligned}$$

Comme φ est transparent, par la proposition 5.14, $\mathrm{eq}_E(\varphi)$ est engendré par les tests $x_j \doteq C_j[M_1, \ldots, M_m]$ et E. Par conséquent, $\mathcal{G} \subseteq \mathrm{Val}'_{A_\eta}(\varphi)$. Réciproquement, (A_η) étant inconditionnellement $=_E$-sûre, les équations de E sont vraies concrètement avec probabilité 1 ; d'où $\mathrm{Val}'_{A_\eta}(\varphi) \subseteq \mathcal{G}$. Ainsi, $\mathcal{G} = \mathrm{Val}'_{A_\eta}(\varphi)$ est précisément l'ensemble des valeurs qui satisfont les tests dans $\mathrm{eq}_E(\varphi)$.

Par hypothèse, $\mathcal{E}$ et $\mathcal{F}$ sont munis des distributions uniformes. Comme α est une bijection de $\mathcal{E}$ vers $\mathcal{G}$, les distributions tirées des sémantiques respectivement concrète et idéale de φ sont donc identiques à la distribution uniforme sur $\mathcal{G}$. □

Toujours dans le cas des distributions uniformes, une conséquence remarquable de la proposition 5.15 est que deux environnements transparents statiquement équivalents sont toujours indistinguables – en fait les sémantiques concrètes sont même identiques.[4] Ceci justifie l'étude de la classe des théories équationnelles transparentes au sens suivant.

Définition 5.7. Une théorie équationnelle E est *transparente* ssi pour tout environnement φ, il existe au moins un environnement transparent $\overline{\varphi}$ tel que $\varphi \approx_E \overline{\varphi}$.

La notion d'environnements transparents est en apparence très voisine de la notion de *motifs* (*patterns*) introduite par M. Abadi et P. Rogaway [AR00] et utilisée dans les travaux postérieurs [MW04a, ABS05] dans le but de définir des équivalences symboliques de messages avec une justification cryptographique. Dans ces travaux, on associe typiquement un motif à chaque message en replacement successivement tout sous-terme déductible par un terme spécial □, signifiant « un certain message indéchiffrable ». Par définition, deux messages sont alors équivalents ss'ils possèdent le même motif (modulo renommage des noms). Par exemple, si $\{M\}_K$ représente le chiffrement probabiliste de M par une clé K, le message $(\{\{K_4\}_{K_3}\}_{K_1}, \{K_1\}_{K_2}, K_2)$ correspond au motif $(\{\Box\}_{K_1}, \{K_1\}_{K_2}, K_2)$. (Comparer avec l'exemple 5.1 où nous avions $\varphi_1 \approx_{E_{\mathsf{enc}}} \overline{\varphi_1}$.)

La notion d'environnement transparent est toutefois définie pour une théorie équationnelle arbitraire. De plus, nous n'excluons pas la possibilité qu'un environnement corresponde à plusieurs environnements transparents. Par exemple, considérons la théorie du OU exclusif (détaillée à la section 6.1) et l'environnement :

$$\varphi = \{x_1 \mapsto n_1 \oplus n_2,\ x_2 \mapsto n_2 \oplus n_3,\ x_3 \mapsto n_1 \oplus n_3\}.$$

Il existe plusieurs environnements transparents équivalents à φ, par exemple $\{x_1 \mapsto n_1 \oplus n_2,\ x_2 \mapsto n_1,\ x_3 \mapsto n_2\}$, $\{x_1 \mapsto n_1,\ x_2 \mapsto n_1 \oplus n_2,\ x_3 \mapsto n_2\}$ et $\{x_1 \mapsto n_1,\ x_2 \mapsto n_2,\ x_3 \mapsto n_1 \oplus n_2\}$.

À l'inverse, la notion d'environnement transparent ne subsume pas celle des motifs (*patterns*) d'Abadi et Rogaway. Notamment, concernant la théorie du chiffrement probabiliste symétrique E_{senc} engendrée par

$$\mathsf{sdec}(\mathsf{senc}(x, y, z), y) \doteq x, \qquad \mathsf{sdec\_success}(\mathsf{senc}(x, y, z), y) \doteq \mathsf{ok}$$

il semble impossible d'associer un environnement transparent à l'environnement $\{x \mapsto \mathsf{senc}(n, k, r), y \mapsto k\}$ $(n, k, r \in \mathcal{N})$, bien que ce dernier soit un motif au sens d'Abadi et Rogaway (adapté à notre syntaxe). En effet, les jetons probabilistes r ne sont pas déductibles, tandis que, intuitivement, le test observable $\mathsf{sdec\_success}(x, y) \doteq \mathsf{ok}$ rend le composant x distinguable d'un nombre purement aléatoire. Exclure les jetons probabilistes de la notion de sous-terme (par exemple ceux-ci sont implicites dans la syntaxe utilisée par Abadi et Rogaway) ne résout pas cette difficulté car les propriétés des environnements transparents élaborées plus

[4] En effet, comme pour le théorème 5.13, deux environnements statiquement équivalents ont la même sémantique idéale par construction.

haut deviennent fausses. Ainsi, les notions de motif et d'environnement transparent nous apparaissent complémentaires.

Outre le résultat de complétude ci-dessous, nous appliquons la notion d'environnements transparents de manière cruciale dans le cas du chiffrement déterministe surjectif (*cipher*) à la section 6.3.

Notons que les propriétés précédentes impliquent la décidabilité de l'équivalence statique pour les théories transparentes dans lesquelles le problème du mot est décidable, pourvu que la construction d'un environnement transparent équivalent à chaque environnement soit effective. En effet, étant donnés deux environnements φ_1 et φ_2, on commence par associer à chacun un environnement transparent équivalent, respectivement $\overline{\varphi_1}$ et $\overline{\varphi_2}$. Il est alors évident de vérifier si $\overline{\varphi_1}$ et $\overline{\varphi_2}$ sont équivalents en utilisant la caractérisation de la proposition 5.14.

Pour finir, nous établissons la complétude de notre critère pour les théories transparentes, dans le cas des probabilités uniformes.

Théorème 5.16. *Soit E une théorie transparente. Soit (A_η) une famille d'algèbres concrètes à distributions uniformes, $\approx_E$-sûre et inconditionnellement $=_E$-sûre. Alors le critère de $\approx_E$-sûreté du théorème 5.13 est satisfait : pour tout environnement φ, $(\llbracket\varphi\rrbracket_{A_\eta}) \approx (\llbracket\varphi\rrbracket^{ideal}_{A_\eta})$.*

Démonstration. Comme E est transparente, il existe un environnement transparent $\overline{\varphi}$ tel que $\varphi \approx_E \overline{\varphi}$. Par $\approx_E$-sûreté, on en déduit $(\llbracket\varphi\rrbracket_{A_\eta}) \approx (\llbracket\overline{\varphi}\rrbracket_{A_\eta})$. Or, la proposition 5.15 implique $(\llbracket\overline{\varphi}\rrbracket_{A_\eta}) = (\llbracket\overline{\varphi}\rrbracket^{ideal}_{A_\eta})$. On en déduit $(\llbracket\varphi\rrbracket_{A_\eta}) \approx (\llbracket\varphi\rrbracket^{ideal}_{A_\eta})$ puisque $\varphi \approx_E \overline{\varphi}$ implique $(\llbracket\overline{\varphi}\rrbracket^{ideal}_{A_\eta}) = (\llbracket\varphi\rrbracket^{ideal}_{A_\eta})$ comme précédemment. □

5.4.3 Cas des probabilités non uniformes

En présence de probabilités non uniformes, la proposition 5.15 n'est pas vraie en général comme le montre l'exemple suivant.

Exemple 5.2. Considérons la théorie équationnelle E_{inv} engendrée par l'équation suivante

$$\mathsf{f}(\mathsf{f}(x)) = x$$

modélisant une certaine fonction involutive $\mathsf{f} : \mathit{Data} \to \mathit{Data}$. Prenons $\llbracket\mathit{Data}\rrbracket_\eta = \{0,1\}^\eta$ et $\llbracket\mathsf{f}\rrbracket_\eta$ égale à la fonction de complémentation bit à bit. On munit $\llbracket\mathit{Data}\rrbracket_\eta$ de la distribution où chaque bit est tiré indépendamment au sort selon une loi de probabilité biaisée : par exemple la probabilité d'un bit 1 est $p = \frac{1}{3}$. Soit n un nom et $\varphi = \{x \mapsto \mathsf{f}(n)\}$. L'environnement φ est transparent. Or $\llbracket\varphi\rrbracket$ et $\llbracket\varphi\rrbracket^{ideal}$ sont facilement distinguables. (En effet, $\llbracket\varphi\rrbracket^{ideal} = \llbracket\{x \mapsto n\}\rrbracket$. Il suffit donc d'observer la proportion de 1 dans les valeurs concrètes.)

Toutefois, il est possible de retrouver un résultat comparable à la proposition 5.15 si l'on se restreint à certains environnements transparents.

Définition 5.8. Un environnement φ est *directement transparent* ss'il vérifie la condition $\text{names}(\varphi) \subseteq \text{im}(\varphi)$. Autrement dit, φ est de la forme

$$\varphi = \{x_1 \mapsto a_1, \ldots, x_m \mapsto a_m, y_1 \mapsto C_1[a_1, \ldots, a_m], \ldots, y_n \mapsto C_n[a_1, \ldots, a_m]\}$$

où $C_1, \ldots, C_n$ sont des contextes publics, et $a_1, \ldots, a_m$ sont des noms mutuellement distincts.

Proposition 5.17. *Soit (A_η) une famille inconditionnellement $=_E$-sûre d'algèbres concrètes. Soit φ un environnement directement transparent. Les sémantiques concrète et idéale de φ coïncident : pour tout η, $[\![\varphi]\!]_{A_\eta} = [\![\varphi]\!]^{ideal}_{A_\eta}$.*

Démonstration. Soit

$$\varphi = \{x_1 \mapsto a_1, \ldots, x_m \mapsto a_m, y_1 \mapsto C_1[a_1, \ldots, a_m], \ldots, y_n \mapsto C_n[a_1, \ldots, a_m]\}$$

Par la proposition 5.14, $\mathrm{eq}(\varphi)$ est engendrée par les équations $y_j \doteq C_i[x_1, \ldots, x_m]$, $1 \leq j \leq n$. L'implémentation étant inconditionnellement $=_E$-sûre, la définition des probabilités conditionnelles de la sémantique idéale implique $[\![\varphi]\!]^{ideal}_{A_\eta} = [\![\varphi]\!]_{A_\eta}$. □

En particulier, étant donné une implémentation inconditionnellement $=_E$-sûre, deux environnements directement transparents et statiquement équivalents sont donc toujours indistinguables. Incidemment, on en déduit comme précédemment que le critère du théorème 5.13 est nécessaire pour les théories équationnelles *directement transparentes*, c'est-à-dire telles que pour tout φ, il existe $\overline{\varphi}$ directement transparent et vérifiant $\varphi \approx_E \overline{\varphi}$.

5.5 Perspectives

Dans ce chapitre nous avons présenté un cadre général — et à notre connaissance original — pour étudier le lien entre modèles logiques et cryptographiques dans le cas passif.

En guise d'application, nous étudions plus particulièrement deux théories équationnelles particulières dans le chapitre suivant (chapitre 6), à savoir : le OU exclusif et une théorie regroupant chiffrements symétrique, asymétrique, et symétrique déterministe surjectif (appelé également cipher ou permutation pseudo-aléatoire).

Naturellement, de nombreuses autres théories méritent d'être étudiées, notamment un regroupement des deux théories précédentes et le cas de l'exponentiation modulaire de type Diffie-Hellman.

Concernant le cadre général développé ici, plusieurs questions ouvertes sont envisageables. D'une part, comme nous l'avons discuté, les notions d'environnements transparents et de motifs (*patterns*) sont incomparables. Du point de vue des applications, une généralisation conjointe ces deux notions serait potentiellement très utile. D'autre part, une question ambitieuse est naturellement la généralisation de ce formalisme au cas actif, c'est-à-dire pour des protocoles en interaction avec l'attaquant.

Chapitre 6

Applications

Sommaire

Ce chapitre a pour but d'appliquer les techniques générales du chapitre précédent (5) à des primitives cryptographiques usuelles.

Nous commençons par traiter le cas du OU exclusif seul (section 6.1). Nous montrons que la modélisation et l'implémentation naturelle de cette primitive satisfont tous les critères de sûreté inconditionnelle et de fidélité du chapitre 5. En particulier, deux environnements statiquement équivalents possèdent donc la même distribution concrète. Ce comportement en un sens idéal du OU exclusif est à rapprocher de la sécurité inconditionnelle du chiffrement One-Time Pad (voir par exemple [ABS05]).

Si les techniques utilisées dans la section 6.1 présentent un intérêt théorique — notamment le traitement algébrique de l'équivalence statique —, peu ou pas de protocoles se limitent au OU exclusif seul. Aussi consacrons-nous l'essentiel des sections suivantes (sections 6.2–6.4) à un jeu de primitives réaliste incluant trois algorithmes chiffrement différents, à savoir

– le chiffrement asymétrique (dit à clés publiques) probabiliste,
– le chiffrement symétrique probabiliste, et

– le chiffrement symétrique déterministe surjectif — appelé également *cipher* ou *permutation pseudo-aléatoire* — en présence éventuelle de clés faibles.

Par clé faible et plus généralement *secret faible*, on entend une donnée à faible entropie, c'est-à-dire pour laquelle l'ensemble des valeurs possibles est à la portée d'une énumération exhaustive (*brute-force testing*). L'exemple le plus manifeste des clés faibles est celui des mots de passe choisis ou mémorisables par un être humain. Nous avons vu dans le chapitre 3 comment modéliser les attaques par dictionnaire sur un secret faible en termes d'(in-)équivalence de processus. La justification de l'équivalence statique pour la théorie considérée ici constitue une justification de la modélisation du chapitre 3 dans le cas particulier des données statiques, modélisées ici sous forme d'environnements ou *frames* (chapitre 5).

La seconde partie de ce chapitre s'articule de la manière suivante. Nous commençons par une étude préliminaire de l'équivalence statique dans un cadre théorique général (section 6.2) afin d'établir plusieurs propriétés techniques utiles par la suite. La section 6.3 détaille le résultat de correction de l'équivalence statique pour les primitives de chiffrement considérées, tandis que la preuve de ce théorème est établie dans la section 6.4. Nous terminons ce chapitre par une comparaison avec les travaux existants (section 6.5).

6.1 Mise en route : le OU exclusif pur

Nous étudions maintenant les problèmes de sûreté et de fidélité définis au chapitre 5 dans le cas de l'implémentation et la modélisation naturelle du OU exclusif, en présence de constantes $0, 1$ et de données purement aléatoires.

Comme modèle symbolique, nous considérons une unique sorte Data, le symbole $\oplus : \mathit{Data} \times \mathit{Data} \to \mathit{Data}$ et les deux constantes $0, 1$, ainsi qu'un nombre infini de noms, notés $a_1, \ldots, a_n, \ldots$ Les termes obtenus sur cette signature sont munis de la théorie équationnelle $E_\oplus$ engendrée par les équations suivantes :

$$\begin{array}{rcl@{\qquad}rcl} (x \oplus y) \oplus z & \doteq & x \oplus (y \oplus z) & x \oplus x & \doteq & 0 \\ x \oplus y & \doteq & y \oplus x & x \oplus 0 & \doteq & x \end{array}$$

En guise d'implémentation, nous définissons les algèbres concrètes A_η, $\eta \geq 0$:
– le domaine concret $[\![\mathit{Data}]\!]_{A_\eta}$ est l'ensemble des chaînes de bits de taille η, $\{0,1\}^\eta$, muni de la distribution uniforme ;
– $\oplus$ est interprété par la fonction XOR habituelle sur $\{0,1\}^\eta$;
– enfin $[\![0]\!]_{A_\eta} = 0^\eta$ et $[\![1]\!]_{A_\eta} = 1^\eta$.

Dans ce formalisme, les environnements statiquement équivalents bénéficient d'une caractérisation algébrique. Soit AC la théorie équationnelle correspondant aux équations d'associativité et de commutativité du symbole $\oplus$ (c'est-à-dire la première colonne ci-dessus). Nous orientons les deux autres équations pour former un système de réécriture $\mathcal{R}_\oplus$ AC-convergent :

$$\begin{array}{rcl} x \oplus x & \to & 0 \\ x \oplus 0 & \to & x \end{array}$$

On montre facilement qu'un terme T est en forme $\mathcal{R}_\oplus/AC$-normale ssi chaque constante et chaque variable apparaît au plus une fois dans T, et 0 n'apparaît pas dans T sauf si $T = 0$.

Soit $a_1, \ldots, a_n$ des noms distincts. En utilisant le système de réécriture $\mathcal{R}_\oplus/AC$, chaque terme clos T avec $\mathrm{names}(T) \subseteq \{a_1, \ldots, a_n\}$ s'écrit $T =_{E_\oplus} \beta_0 \oplus \bigoplus_{j=1}^{n} \beta_j\, a_j$ où $\beta_j \in \{0,1\}$, les a_j sont mutuellement distincts, et l'on utilise la convention $0a_j = 0$ et $1a_j = a_j$. Dans la suite, on voit $\{0,1\}$ comme le corps à deux éléments $\mathbb{F}_2$; ainsi les termes modulo $=_{E_\oplus}$ forment un $\mathbb{F}_2$-espace vectoriel.

De la même manière un environnement φ tel que $\mathrm{names}(\varphi) \subseteq \{a_1, \ldots, a_n\}$ s'écrit

$$\varphi =_{E_\oplus} \left\{ x_1 \mapsto \alpha_{1,0} \oplus \bigoplus_{j=1}^{n} \alpha_{1,j}\, a_j \ , \ \ldots \ , \ x_m \mapsto \alpha_{m,0} \oplus \bigoplus_{j=1}^{n} \alpha_{m,j}\, a_j \right\}$$

où $\alpha_{i,j} \in \mathbb{F}_2$. Groupons les coefficients $\alpha_{i,j}$ dans une matrice de taille $(m+1)\times(n+1)$ sur à coefficients dans $\mathbb{F}_2$. Alors φ est décrite par la relation formelle

$$\begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_m \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & \ldots & 0 \\ \alpha_{1,0} & \alpha_{1,1} & \ldots & \alpha_{1,n} \\ \vdots & & & \vdots \\ \alpha_{m,0} & \alpha_{m,1} & \ldots & \alpha_{m,n} \end{pmatrix}}_{\alpha} \cdot \begin{pmatrix} 1 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}$$

Nous donnons maintenant une caractérisation de l'ensemble $\mathrm{eq}_{E_\oplus}(\varphi)$ des tests valides dans φ. Soit M, N deux termes tels $\mathrm{var}(M,N) \subseteq \mathrm{dom}(\varphi)$ et $\mathrm{names}(M,N) \cap \mathrm{names}(\varphi) = \emptyset$. Dans la mesure où $M\varphi =_{E_\oplus} N\varphi$ équivaut à $(M \oplus N)\varphi =_{E_\oplus} 0$, on peut supposer $N = 0$.

Supposons M en forme normale. Comme $M\varphi =_{E_\oplus} 0$ et $\mathrm{names}(M) \cap \mathrm{names}(\varphi) = \emptyset$, on en déduit $\mathrm{names}(M) = \emptyset$. Soit $M =_{AC} \beta_0 \oplus \bigoplus_{i=1}^{m} \beta_i\, x_i$. La condition $M\varphi =_{E_\oplus} 0$ équivaut à l'équation vectorielle

$$(\beta_0, \ldots, \beta_m) \cdot \alpha = 0$$

autrement dit, $(\beta_0, \ldots, \beta_m)$ appartient au co-noyau de α, noté $\mathrm{coker}(\alpha)$.

Finalement, soit φ, φ' deux environnements tels que $\mathrm{names}(\varphi, \varphi') \subseteq \{a_1, \ldots, a_n\}$ et $\mathrm{dom}(\varphi) = \mathrm{dom}(\varphi') = \{x_1, \ldots, x_m\}$. Soit α et α' les $(m+1) \times (n+1)$-matrices correspondantes définies comme ci-dessus. De la discussion précédente, on déduit

$$\varphi \approx_{E_\oplus} \varphi' \ \Leftrightarrow \ \mathrm{coker}(\alpha) = \mathrm{coker}(\alpha')$$

c'est-à-dire, si nous écrivons $\mathrm{im}(\alpha) = \{\alpha \cdot \gamma \,|\, \gamma \in (\mathbb{F}_2)^{(n+1)\times 1}\}$ l'image de α, nous avons par dualité

$$\varphi \approx_{E_\oplus} \varphi' \ \Leftrightarrow \ \mathrm{im}(\alpha) = \mathrm{im}(\alpha'). \tag{6.1.1}$$

Cette caractérisation est un ingrédient important du résultat suivant sur la théorie du XOR.

Théorème 6.1. *L'implémentation du XOR pour la signature considérée, notée ci-dessus (A_η), est inconditionnellement $=_{E_\oplus}$-, $\approx_{E_\oplus}$- et $\not\vdash_{E_\oplus}$-sûre. Elle est également $=_{E_\oplus}$-, $\approx_{E_\oplus}$- et $\not\vdash_{E_\oplus}$-fidèle.*

Démonstration. La $=_E$-sûreté inconditionnelle est claire. On en déduit la $\not\vdash_{E_\oplus}$-fidélité par proposition 5.2.

Montrons que (A_η) est $=_{E_\oplus}$-fidèle. Sachant que $T_1 \neq_{E_\oplus} T_2$ équivaut à $T_1 \oplus T_2 \neq_{E_\oplus} 0$, on considère seulement le cas d'un terme T clos, normal tel que

$T \neq 0$. Si $T = 1$, la sémantique concrète de T est la constante 1^η. Dans tous les autres cas, il s'agit de la distribution uniforme sur $\{0,1\}^\eta$. Ainsi $\mathbb{P}\left[[\![T]\!]_{A_\eta} = 0\right]$ est négligeable et l'implémentation est donc $=_{E_\oplus}$-fidèle. Par la proposition 5.2, on en déduit la $\approx_{E_\oplus}$-fidélité.

Nous montrons maintenant la $\approx_{E_\oplus}$-sûreté inconditionnelle. Soit φ un environnement tel que $\mathrm{names}(\varphi) \subseteq \{a_1, \ldots, a_n\}$, et $\alpha = (\alpha_{i,j})$ sa $(m+1)\times(n+1)$-matrice associée comme précédemment. On voit α comme une fonction $\mathbb{F}_2$-linéaire de $(\mathbb{F}_2)^{n+1}$ vers $(\mathbb{F}_2)^{m+1}$.

Pour des raisons de simplicité, fixons l'ordre des variables dans $\mathrm{dom}(\varphi)$, et assimilons $\mathrm{Val}_{A_\eta}(\varphi)$, l'ensemble des valeurs concrètes de φ, à l'ensemble $\mathcal{F} = \{1^\eta\} \times (\mathbb{F}_2)^{m\eta}$ où les η premiers bits à 1 sont ajoutés pour des raisons techniques.

La sémantique concrète usuelle de φ consiste en un tirage uniforme d'un vecteur de $\mathcal{E} = \{1^\eta\} \times (\mathbb{F}_2)^{n\eta}$ (pour les valeurs des noms), auquel on applique une certaine fonction $\mathbb{F}_2$-linéaire $\widehat{\alpha} : (\mathbb{F}_2)^{(n+1)\eta} \to (\mathbb{F}_2)^{(m+1)\eta}$. Plus précisément, si l'on voit $(\mathbb{F}_2)^{(n+1)\eta}$ comme le produit $\underbrace{{\mathbb{F}_2}^\eta \times \cdots \times {\mathbb{F}_2}^\eta}_{n+1}$ et de même $(\mathbb{F}_2)^{(m+1)\eta} \simeq \underbrace{{\mathbb{F}_2}^\eta \times \cdots \times {\mathbb{F}_2}^\eta}_{m+1}$, la fonction $\widehat{\alpha}$ est définie par

$$\widehat{\alpha}\,(f_0, \ldots, f_n) = \left(\bigoplus_{j=0}^{n} \alpha_{0,j}\, f_j, \ldots, \bigoplus_{j=0}^{n} \alpha_{m,j}\, f_j \right)$$

Comme $\widehat{\alpha}$ est linéaire, toutes les images inverses $\widehat{\alpha}^{-1}(\{x\})$, $x \in \mathrm{im}(\widehat{\alpha})$, ont le même nombre d'éléments. Par conséquent, la sémantique concrète de φ est également la distribution uniforme sur $\widehat{\alpha}(\mathcal{E}) = \mathrm{im}(\widehat{\alpha}) \cap \mathcal{F}$.

Soit φ' un environnement tel que $\varphi \approx_{E_\oplus} \varphi'$ et $\mathrm{names}(\varphi') \subseteq \{a_1, \ldots, a_n\}$. Soit α' et $\widehat{\alpha'}$ définis comme ci-dessus. L'équation 6.1.1 implique $\mathrm{im}(\alpha) = \mathrm{im}(\alpha')$. Or, si l'on voit $(\mathbb{F}_2)^{(m+1)\eta}$ comme $\underbrace{{\mathbb{F}_2}^{m+1} \times \cdots \times {\mathbb{F}_2}^{m+1}}_{\eta}$, on a $\widehat{\alpha} = \underbrace{\alpha \times \cdots \times \alpha}_{\eta}$ et de même $\widehat{\alpha'} = \underbrace{\alpha' \times \cdots \times \alpha'}_{\eta}$. Par conséquent,

$$\mathrm{im}(\widehat{\alpha}) = \underbrace{\mathrm{im}(\alpha) \times \cdots \times \mathrm{im}(\alpha)}_{\eta} = \underbrace{\mathrm{im}(\alpha') \times \cdots \times \mathrm{im}(\alpha')}_{\eta} = \mathrm{im}(\widehat{\alpha'})$$

ce qui implique que φ et φ' ont la même sémantique concrète. Ainsi (A_η) est donc inconditionnellement $\approx_{E_\oplus}$-sûre.

Pour finir, nous prouvons la $\not\approx_{E_\oplus}$-sûreté inconditionnelle. Soit φ un environnement et T un terme, tous deux bien formés et en forme normale, tels que $\varphi \not\approx_{E_\oplus} T$ et $\mathrm{names}(T) \subseteq \mathrm{names}(\varphi) = \{a_1, \ldots, a_n\}$. Soit α associée à φ comme précédemment et $T =_{AC} \beta_0 \oplus \bigoplus_{j=1}^{n} \beta_j\, a_j$.

Soit γ la $(m+2)\times(n+1)$-matrice obtenue en ajoutant à α une dernière ligne égale à $\beta = (\beta_0, \ldots, \beta_n)$:

$$\gamma = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ \alpha_{1,0} & \alpha_{1,1} & \ldots & \alpha_{1,n} \\ \vdots & & & \vdots \\ \alpha_{m,0} & \alpha_{m,1} & \ldots & \alpha_{m,n} \\ \beta_0 & \beta_1 & \ldots & \beta_{n_0} \end{pmatrix}$$

Comme $\varphi \not\vdash_{E_\oplus} T$, il n'existe pas de terme M tel que $\mathrm{var}(M) \subseteq \mathrm{dom}(\varphi)$, $\mathrm{names}(M) \cap \mathrm{names}(\varphi, T) = \emptyset$ et $M\varphi =_{E_\oplus} T$. En particulier, β est linéairement indépendante des autres lignes de γ.

Notamment, β est indépendante de la première ligne $(1, 0, \ldots, 0)$, donc il existe $j \geq 1$ tel que $\beta_j \neq 0$. On en déduit que la distribution $(\xleftarrow{R} [\![T]\!]_{A_\eta})$ est la distribution uniforme sur $\{0,1\}^\eta$ (par conséquent sans collisions).

Concernant la première condition de la $\not\vdash_E$-sûreté inconditionnelle (indépendance des tirages de φ et T), on montre comme précédemment que la sémantique concrète de (φ, T) est la distribution uniforme sur l'image de $\mathcal{E} = \{1^\eta\} \times (\mathbb{F}_2)^{n\eta}$ par la fonction $\widehat{\gamma}$ (définie à partir de γ de manière similaire à $\widehat{\alpha}$ ci-dessus). Nous voyons β comme une fonction linéaire de $(\mathbb{F}_2)^{n+1}$ vers $\mathbb{F}_2$ et définissons $\widehat{\beta}$ comme précédemment. Dans la suite, nous prouvons que l'image $\gamma(\mathcal{E})$ est le produit cartésien des deux ensembles $\widehat{\alpha}(\mathcal{E})$ et $\widehat{\beta}(\mathcal{E})$. Il s'ensuit que les tirages de φ et T sont indépendants.

L'inclusion $\widehat{\gamma}(\mathcal{E}) \subseteq \widehat{\alpha}(\mathcal{E}) \times \widehat{\beta}(\mathcal{E})$ est évidente. Inversement, comme β est indépendant des lignes de α, il existe un vecteur $u \in (\mathbb{F}_2)^{n+1}$ tel que $\beta(u) = 1$ et $\alpha(u) = 0$ (sinon $\ker(\beta) \supseteq \ker(\alpha)$ implique $\beta \in \mathrm{coim}(\beta) \subseteq \mathrm{coim}(\alpha)$). Soit $x, y \in \mathcal{E}$. Nous montrons qu'il existe $z \in \mathcal{E}$ tel que $\widehat{\alpha}(z) = \widehat{\alpha}(x) \in (\mathbb{F}_2)^{(m+1)\eta}$ et $\widehat{\beta}(z) = \widehat{\beta}(y) \in (\mathbb{F}_2)^\eta$.

En effet, voyons $\mathcal{E}$ comme $(\{1\} \times (\mathbb{F}_2)^n)^\eta$. En utilisant les bases correspondantes, soit $x = (x_1, \ldots, x_\eta)$ et $y = (y_1, \ldots, y_\eta)$ avec $x_i, y_i \in \{1\} \times (\mathbb{F}_2)^n$. Soit $z_i = x_i + (\beta(y_i) - \beta(x_i)) \cdot u$ et $z = (z_1, \ldots, z_\eta)$. Ainsi, $\widehat{\alpha}(z) = (\alpha(z_1), \ldots, \alpha(z_\eta)) = (\alpha(x_1), \ldots, \alpha(x_\eta)) = \widehat{\alpha}(x)$ et $\widehat{\beta}(z) = (\beta(z_1), \ldots, \beta(z_\eta)) = (\beta(y_1), \ldots, \beta(y_\eta)) = \widehat{\beta}(y)$. De plus, $\alpha(u) = 0$ implique que la première coordonnée de u est 0, ainsi la première coordonnée de chaque z_i est 1, c'est-à-dire, $z \in \mathcal{E}$. □

Nous concluons cette section par une preuve du fait que $E_\oplus$ est transparente comme annoncé dans la section 5.4.

Proposition 6.2. *La théorie $E_\oplus$ est transparente : pour tout environnement φ, il existe un environnement transparent $\overline{\varphi}$ tel que $\varphi \approx_{E_\oplus} \overline{\varphi}$.*

Démonstration. Soit φ un environnement et α sa matrice $(m+1) \times (n+1)$ associée comme précédemment. Soit d la dimension de $\mathrm{im}(\alpha)$ (*i.e.* le rang de α). Il existe une sous-matrice α' de taille $(m+1) \times d$ de α telle que α' est injective et $\mathrm{im}(\alpha') = \mathrm{im}(\alpha)$ (considérer un sous-ensemble maximal de colonnes de α). Comme la première colonne de α est indépendante des autres (son premier élément est 1 tandis que les autres colonnes commencent par 0), on peut supposer sans perte de généralité que $d \geq 1$ et que la première colonne de α' est celle de α.

Soit $a'_1, \ldots, a'_{d-1}$ des noms distincts. Soit φ' l'environnement associé à la matrice α', selon la relation

$$\begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_m \end{pmatrix} = \alpha' \cdot \begin{pmatrix} 1 \\ a'_1 \\ \vdots \\ a'_{d-1} \end{pmatrix}.$$

Comme $\mathrm{im}(\alpha') = \mathrm{im}(\alpha)$, on a $\varphi' \approx_{E_\oplus} \varphi$. De plus, comme α' est injective, il existe α'' tel que $\alpha'' \cdot \alpha'$ est la matrice identité $d \times d$. Ceci implique que tous les a'_i sont déductible de φ', autrement dit, φ' est transparente. □

6.2 Étude préliminaire de l'équivalence statique

Dans cette section nous étudions certaines propriétés de l'équivalence statique en présence de symboles de chiffrement et d'une théorie équationnelle disjointe arbitraire. Notamment, sous des hypothèses générales, nous montrons qu'il est correct vis-à-vis de l'équivalence statique de remplacer un sous-terme représentant un message chiffré par un nom frais lorsque la clé de déchiffrement correspondante n'est pas déductible.

Ce type de raisonnement — par modifications successives du problème de départ — est courant dans le modèle cryptographique [Sho04]. Le résultat principal de cette section (proposition 6.4) permet d'appliquer cette méthode également dans le modèle symbolique. Il s'agit d'un ingrédient important de la preuve du théorème de $\approx_E$-sûreté de la section 6.3.

Rappelons qu'un système de réécriture $\mathcal{R}$ a la propriété *sous-terme* (chapitre 2) ssi pour tout règle $l \to r$ dans $\mathcal{R}$, r est ou bien (i) un sous-terme strict de l, ou bien (ii) un terme $\mathcal{R}$-normal tel que $\mathrm{var}(r) = \emptyset$. Par la proposition 2.8, on sait qu'un tel système est toujours terminant (*i.e.* nœthérien).

Nous commençons par établir un lemme d'interpolation généralisant la caractérisation des symboles libres décrite au chapitre 2 (théorème 2.2).

Lemme 6.3. *Soit $\mathcal{F}_1$ et $\mathcal{F}_2$ deux ensembles de symboles disjoints et $\mathcal{F} = \mathcal{F}_1 \uplus \mathcal{F}_2$. Soit $\mathcal{R}_1$ et $\mathcal{R}_2$ deux systèmes de réécriture sur $\mathcal{F}[\mathcal{X}]$ vérifiant les conditions suivantes :*

(i) $\mathcal{R}_1$ est sous-terme ;

(ii) pour toute règle $l \to r \in \mathcal{R}_1$, le symbole de tête de l appartient à $\mathcal{F}_1$;

(iii) pour toute règle $l \to r \in \mathcal{R}_2$, l et r sont dans $\mathcal{F}_2[\mathcal{X}]$;

(iv) $\mathcal{R}_1 \cup \mathcal{R}_2$ est confluent ;

(v) $\mathcal{R}_1 = \emptyset$ ou bien pour chaque règle $l \to r \in \mathcal{R}_2$, $\mathrm{var}(r) \subseteq \mathrm{var}(l)$.

Soit $T_0 = h(U_1, \ldots, U_{n_0})$ un terme de $\mathcal{F}[\mathcal{X}]$ tel que $h \in \mathcal{F}_1$ et que pour toute règle $l \to r \in \mathcal{R}_1$ du second type (i.e. lorsque r n'est pas sous-terme strict de l), h n'apparaît pas dans r.

Notons $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ et E la théorie équationnelle engendrée par $\mathcal{R}$. Soit x_0 une variable quelconque de la même sorte que T_0, et $\rho = \{T_0 \mapsto_E x_0\}$ le remplacement modulo E envoyant T_0 sur x_0 (voir chapitre 2).

Soit T_1 et T_2 deux termes vérifiant les conditions suivantes :

(a) $T_1 =_E T_2$,

(b) pour tout sous-terme non variable $S = f(S_1, \ldots, S_n)$ de T_1 ou T_2, pour toute règle $l \to r \in \mathcal{R}_1$ telle que $l = f(l_1, \ldots, l_n)$, pour toute substitution θ telle que $\forall j, S_j \to^*_{\mathcal{R}} l_j\theta$, $(l\theta)\rho = l(\theta\rho)$.

Alors, on a l'égalité $T_1\rho =_E T_2\rho$.

En pratique, on vérifiera l'égalité $(l\theta)\rho = l(\theta\rho)$ de la condition (b) en s'assurant que ρ n'opère pas aux positions non variables de l dans $l\theta$:

pour tout sous-terme $t = h(t_1, \ldots, t_{n_0})$ de l, il existe i tel que $U_i \neq_E t_i\theta$.

Démonstration. Soit $\mathcal{P}(T)$ la propriété suivante :

> Pour tout sous-terme non variable $S = f(S_1, \ldots, S_n)$ de T, pour toute règle $l \to r \in \mathcal{R}_1$ telle que $l = f(l_1, \ldots, l_n)$, pour toute substitution θ telle que $\forall j, S_j \to^*_{\mathcal{R}} l_j\theta$, $(l\theta)\rho = l(\theta\rho)$.

La condition (b) est donc équivalente à $\mathcal{P}(T_1)$ et $\mathcal{P}(T_2)$.

Par la condition (iv), $T_1 =_E T_2$ implique l'existence d'une chaîne

$$T_1 = T^0 \to^*_{\mathcal{R}_2} \to_{\mathcal{R}_1} \to^*_{\mathcal{R}_2} T^1 \ldots \leftarrow^*_{\mathcal{R}_2} \leftarrow_{\mathcal{R}_1} \leftarrow^*_{\mathcal{R}_2} T^m = T_2$$

Pour commencer, nous montrons la propagation de $\mathcal{P}$ le long de la chaîne entre T_1 et T_2. Plus précisément, nous établissons les faits suivants :

1. $\mathcal{P}(T)$ et $T \to_{\mathcal{R}_1} T'$ impliquent $\mathcal{P}(T')$;
2. $\mathcal{P}(T)$ et $T \to_{\mathcal{R}_2} T'$ impliquent $\mathcal{P}(T')$.

Si $\mathcal{R}_1 = \emptyset$ alors $\mathcal{P}(T)$ est toujours vraie et les deux implications sont valides. Dans le cas contraire, nous raisonnons comme suit :

1. Supposons $\mathcal{P}(T)$ et $T \to_{\mathcal{R}_1} T'$. Il existe une position p, une règle $l \to r \in \mathcal{R}_1$, et une substitution σ telles que $T|_p = l\sigma$ et $T' = T[r\sigma]_p$.
   Supposons qu'il existe un sous-terme $S = f(S_1, \ldots, S_n)$ de T', une règle $l' = f(l_1, \ldots, l_n) \to r' \in \mathcal{R}_1$, et une substitution θ tels que $\forall i, S_i \to^*_{\mathcal{R}} l_i\theta$.
   Nous montrons que $(l'\theta)\rho = l'(\theta\rho)$ en distinguant trois cas.
   (a) Si S est un sous-terme de $r\sigma$, alors par la condition (i), r est un sous-terme strict de l (si r est clos et $\mathcal{R}_1$-normal, $S = l'\theta$ ne peut pas être un sous-terme de $r\sigma = r$), donc S est un sous-terme de $l\sigma$. Par conséquent, on conclut directement par $\mathcal{P}(T)$.
   (b) Si S est un sous-terme de $T'[\_]_p = T[\_]_p$, alors $\mathcal{P}(T)$ s'applique également.
   (c) Sinon, S s'écrit $S = T|_{q_1}[r\sigma]_{q_2}$ où $p = q_1q_2$ et $q_1 \neq \epsilon$. Il existe donc $S'_1, \ldots, S'_n$ tels que $T|_{q_1} = T|_{q_1}[l\sigma]_{q_2} = f(S'_1, \ldots, S'_n)$ et $\forall i, S'_i \to^*_{\mathcal{R}} S_i \to^*_{\mathcal{R}} l_i\theta$. On applique alors $\mathcal{P}(T)$ à $T|_{q_1} = f(S'_1, \ldots, S'_n)$, $l \to r$ et θ.
2. Supposons $\mathcal{P}(T)$ et $T \to_{\mathcal{R}_2} T'$. Alors il existe une position p, une règle $l \to r \in \mathcal{R}_2$, et une substitution σ telle que $T|_p = l\sigma$ et $T' = T[r\sigma]_p$.
   Supposons qu'il existe un sous-terme $S = f(S_1, \ldots, S_n)$ de T', une règle $l' = f(l_1, \ldots, l_n) \to r' \in \mathcal{R}_1$, et une substitution θ tels que $\forall i, S_i \to^*_{\mathcal{R}} l_i\theta$.
   Nous montrons que $(l'\theta)\rho = l'(\theta\rho)$ en distinguant à nouveau trois cas.
   (a) Si S est un sous-terme de $r\sigma$, alors, comme par la condition (ii), $f \notin \mathcal{F}_2$, S est un sous-terme de $\mathrm{var}(r)\sigma$. Par la condition (v), S est donc un sous-terme de $l\sigma$. Par conséquent, $\mathcal{P}(T)$ s'applique.
   (Les deux cas restants se traitent comme ci-dessus.)

Des points précédents et des hypothèses $\mathcal{P}(T_1)$ et $\mathcal{P}(T_2)$, on déduit que $\mathcal{P}$ est vraie sur tous les termes intermédiaires de la chaîne ci-dessus. Par conséquent, pour conclure la preuve, il suffit de vérifier les deux points suivants :

1. $\mathcal{P}(T)$ et $T \to_{\mathcal{R}_1} T'$ impliquent $T\rho =_E T'\rho$;
2. $T \to_{\mathcal{R}_2} T'$ implique $T\rho =_E T'\rho$.

Nous procédons comme suit.

1. Supposons $\mathcal{P}(T)$ et $T \to_{\mathcal{R}_1} T'$. Il existe une position p, une règle $l \to r \in \mathcal{R}_1$, et une substitution σ telles que $T|_p = l\sigma$ et $T' = T[r\sigma]_p$.
   Si p est situé strictement en-dessous des positions du symbole h où ρ opère dans T, alors par définition de l'application de ρ, sachant que $T \xrightarrow{p}_{\mathcal{R}_1} T'$, on a $T\rho = T'\rho$.
   Dans le cas contraire, on a $(T\rho)|_p = (T|_p)\rho = (l\sigma)\rho = l(\sigma\rho)$ par $\mathcal{P}(T)$ appliqué à $T|_p = l\sigma$. De plus, comme r est ou bien un sous-terme de l ou bien un terme clos ne contenant pas h, on a également $(r\sigma)\rho = r(\sigma\rho)$. D'où $T'\rho = (T\rho)[(r\sigma)\rho]_p = (T\rho)[r(\sigma\rho)]_p$ et par conséquent $T\rho =_E T'\rho$.
2. Supposons $T \to_{\mathcal{R}_2} T'$. Il existe une position p, une règle $l \to r \in \mathcal{R}_2$, et une substitution σ telles que $T|_p = l\sigma$ et $T' = T[r\sigma]_p$.
   Si p est situé strictement en dessous des positions du symbole h où ρ opère dans T, alors $T \xrightarrow{p}_{\mathcal{R}_2} T'$ implique $T\rho = T'\rho$.
   Dans le cas contraire, notons que par la condition (iii), et du fait que $h \notin \mathcal{F}_2$, on a $(l\sigma)\rho = l(\sigma\rho)$ et $(r\sigma)\rho = r(\sigma\rho)$. Par conséquent, $(T\rho)|_p = (T|_p)\rho = (l\sigma)\rho = l(\sigma\rho)$ et $T'\rho = (T\rho)[(r\sigma)\rho]p = (T\rho)[r(\sigma\rho)]_p$ et finalement $T\rho =_E T'\rho$. □

En guise d'application, nous étudions maintenant l'équivalence statique en présence de chiffrement pour des clés non déductibles. Le résultat obtenu s'appliquera en particulier à la théorie utilisée à la section 6.3.

Proposition 6.4. *Soit $\mathcal{F} = \mathcal{F}_{pub} \uplus \mathcal{N}$ un ensemble de symboles partitionné en symboles publics et en noms. On considère les systèmes de réécriture, les termes et les sous-ensembles de symboles suivants :*

$$
\begin{array}{rcl}
\mathcal{R}_1 & = & \{\, \mathsf{dec}(\mathsf{enc}(x,y),y) \to x, \quad \mathsf{enc}(\mathsf{dec}(x,y),y) \to x \,\} \\
T_1 & = & \mathsf{enc}(U_1, U_2) \\
\mathcal{F}_1 & = & \{\mathsf{enc}, \mathsf{dec}\}
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{R}_2 & = & \{\, \mathsf{sdec}(\mathsf{senc}(x,y,z),y) \to x, \quad \mathsf{sdec\_success}(\mathsf{senc}(x,y,z),y) \to 1 \,\}, \\
T_2 & = & \mathsf{senc}(U_1, U_2, U_3) \\
\mathcal{F}_2 & = & \{\mathsf{senc}, \mathsf{sdec}, \mathsf{sdec\_success}\}
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{R}_3 & = & \{\, \mathsf{pdec}(\mathsf{penc}(x, \mathsf{pub}(y), z), y) \to x, \\
 & & \quad \mathsf{pdec\_success}(\mathsf{penc}(x, \mathsf{pub}(y), z), y) \to 1 \,\} \\
T_3 & = & \mathsf{penc}(U_1, U_2, U_3) \\
\mathcal{F}_3 & = & \{\mathsf{penc}, \mathsf{pdec}, \mathsf{pdec\_success}\}
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{R}_4 & = & \mathcal{R}_3 \\
T_4 & = & \mathsf{pub}(U_2) \\
\mathcal{F}_4 & = & \mathcal{F}_3
\end{array}
$$

Notons que pour chaque i, $\mathcal{R}_i$ est sous-terme et convergent.

Fixons $i \in \{1,2,3,4\}$. Soit $\mathcal{F}_0 = \mathcal{F}_{pub} - \mathcal{F}_i$. Soit $\mathcal{R}_0$ un système de réécriture vérifiant les points suivants :

(iii) les règles de $\mathcal{R}_0$ sont faites de symboles dans $\mathcal{F}_0$;

(iv) $\mathcal{R}_i \cup \mathcal{R}_0$ *est confluent ;*

(v) pour chaque règle $l \to r \in \mathcal{R}_0$, $\mathrm{var}(r) \subseteq \mathrm{var}(l)$.

Soit $\mathcal{R} = \mathcal{R}_i \cup \mathcal{R}_0$ *et* E *la théorie équationnelle associée à* $\mathcal{R}$. *Notons* $\to_{\mathcal{R}_i/\mathcal{R}_0}$ *la relation* $\to^*_{\mathcal{R}_0} \to_{\mathcal{R}_i} \to^*_{\mathcal{R}_0}$.

Soit φ *un environnement en forme* $\mathcal{R}_i/\mathcal{R}_0$*-normale — autrement dit, il n'existe pas de* $x \in \mathrm{dom}(\varphi)$ *et de* T' *tel que* $x\varphi \to^*_{\mathcal{R}_0} \to_{\mathcal{R}_i} \to^*_{\mathcal{R}_0} T'$. *Supposons que les conditions suivantes sont vérifiées :*

- *si* $i \in \{1, 2, 4\}$, *alors* $\varphi \not\vdash_E U_2$ *;*
- *si* $i = 3$, *alors*
  - *il existe* $j \in \{1, 2, 3\}$ *tel que* $\varphi \not\vdash_E U_j$, *et*
  - *pour tout* V *tel que* $U_2 =_E \mathsf{pub}(V)$, $\varphi \not\vdash_E V$.

Soit a *un nom frais. Alors, on a l'équivalence*

$$\varphi \approx_E \varphi\{T_i \mapsto_E a\}.$$

Démonstration. Soit ρ le remplacement $\{T_i \mapsto a\}$, et $\varphi' = \varphi\rho$.

Si M et N sont deux termes tels que $\mathrm{var}(M, N) \subseteq \mathrm{dom}(\varphi)$, $\mathrm{names}(M, N) \cap \mathrm{names}(\varphi) = \emptyset$ et $M\varphi' =_E N\varphi'$, alors par stabilité de E par remplacement des noms (proposition 2.5), $M\varphi =_E N\varphi$.

Réciproquement, supposons que $M\varphi =_E N\varphi$ avec $\mathrm{names}(M, N) \cap \mathrm{names}(\varphi) = \emptyset$. Nous prouvons successivement

1. $(M\varphi)\rho = M(\varphi\rho)$ et $(N\varphi)\rho = N(\varphi\rho)$,
2. la condition (b) du lemme 6.3 est satisfaite sur $M\varphi$ et $N\varphi$.

En utilisant le lemme 6.3, on obtient alors $M\varphi' = (M\varphi)\rho =_E (N\varphi)\rho = N\varphi'$.

Nous considérons uniquement le cas de M dans la mesure où celui de N est identique. Notons $T_i = h(U_1, \ldots, U_{n_0})$.

1. Si $(M\varphi)\rho \neq M(\varphi\rho)$, alors il existe une position $p \in \mathrm{pos}(M)$ telle que $M|_p$ est de la forme $M|_p = h(M_1, \ldots, M_{n_0})$ et $\forall j, U_j =_{E_0} M_j\varphi$. En particulier, $\forall j, \varphi \vdash_E U_j$; contradiction.
2. Soit $S = (M\varphi)|_p = f(S_1, \ldots, S_n)$ un sous-terme de $M\varphi$ en position p, $l = f(l_1, \ldots, l_n) \to r$ une règle de $\mathcal{R}_i$ et θ une substitution tels que $\forall j, S_j \to^*_{\mathcal{R}} l_j\theta$. Montrons que $(l\theta)\rho = l(\theta\rho)$.

   Comme φ est en forme $\mathcal{R}_i/\mathcal{R}_0$-normale et que $S \to^*_{\mathcal{R}} l\theta \to_{\mathcal{R}_i} r\theta$, p est une position non variable de M. Ainsi, il existe un sous-terme $f(M_1, \ldots, M_n) = M|_p$ de M tel que $\forall j, M_j\varphi = S_j \to^*_{\mathcal{R}} l_j\theta$.

   Nous vérifions que

   > pour tout sous-terme $t = h(t_1, \ldots, t_{n_0})$ de l, il existe i tel que $U_i \neq_E t_i\theta$.

   en distinguant plusieurs cas selon la valeur de i et la règle $l \to r \in \mathcal{R}_i$.

   (a) Si $i = 1$,
   - Si $l = \mathsf{dec}(\mathsf{enc}(x, y), y)$, le seul sous-terme à considérer est $t = \mathsf{enc}(x, y)$. Or, $t_2 = y = l_2$ vérifie $U_2 \neq_E t_2\theta$. En effet, dans le cas contraire, on en déduit $U_2 =_E l_2\theta \leftarrow^*_{\mathcal{R}} M_2\varphi$ ce qui contredit $\varphi \not\vdash_E U_2$.
   - Si $l = \mathsf{enc}(\mathsf{dec}(x, y), y)$, le seul sous-terme à considérer est $t = l$. Comme ci-dessus, $t_2 = y = l_2$ vérifie $U_2 \neq_E t_2\theta$.

(b) Si $i = 2$,
- Si $l = \mathsf{sdec}(\mathsf{senc}(x, y, z), y)$, le seul sous-terme à considérer est $t = \mathsf{senc}(x, y, z)$. À nouveau, $t_2 = y = l_2$ satisfait $U_2 \neq_E t_2\theta$.
- Le cas $l = \mathsf{sdec\_success}(\mathsf{senc}(x, y, z), y)$ est similaire.

(c) Si $i = 3$,
- Si $l = \mathsf{pdec}(\mathsf{penc}(x, \mathsf{pub}(y), z), y)$, le seul sous-terme à considérer est $t = \mathsf{penc}(x, \mathsf{pub}(y), z)$. Le terme $t_2 = \mathsf{pub}(y) = \mathsf{pub}(l_2)$ vérifie $U_2 \neq_E t_2\theta$. En effet, dans le cas contraire, on a $U_2 =_E \mathsf{pub}(l_2\theta)$ et $l_2\theta \leftarrow^*_{\mathcal{R}} M_2\varphi$, ce qui contredit l'hypothèse $\forall V, U_2 =_E \mathsf{pub}(V) \Rightarrow \varphi \not=_E V$.
- le cas $l = \mathsf{pdec\_success}(\mathsf{penc}(x, \mathsf{pub}(y), z), y)$ est similaire.

(d) Si $i = 4$,
- Si $l = \mathsf{pdec}(\mathsf{penc}(x, \mathsf{pub}(y), z), y)$, le seul sous-terme à considérer est $t = \mathsf{pub}(y)$. Comme précédemment, $t_1 = y = l_2$ vérifie $U_2 \neq_E t_1\theta$.
- Le cas $l = \mathsf{pdec\_success}(\mathsf{penc}(x, \mathsf{pub}(y), z), y)$ est identique. □

Remarque. Notons que les conditions (iii)-(v) imposées à $\mathcal{R}_0$ dans la proposition ci-dessus sont très générales. En particulier, $\mathcal{R}_0$ peut être le symétrisé d'un ensemble d'équations E_0 :

$$\mathcal{R}_0 = \mathcal{R}(E_0) = \{l \to r, r \to l \mid l \doteq r \in E_0\}$$

Dans ce cas les conditions sur $\mathcal{R}_0$ deviennent

(iii) les équations de $\mathcal{E}_0$ sont faites de symboles dans $\mathcal{F}_0 = \mathcal{F}_{\text{pub}} - \mathcal{F}_i$;

(iv) $\mathcal{R}_i$ est E_0-confluent ;

(v) pour chaque équation $l \doteq r \in \mathcal{R}_0$, $\text{var}(l) = \text{var}(r)$;

C'est le cas par exemple pour la présentation usuelle de la théorie du OU exclusif (section 6.1).

6.3 Chiffrement, clés faibles et attaques par dictionnaire

Nous montrons maintenant la correction de l'équivalence statique pour un ensemble de primitives cryptographiques comprenant :
- la construction de paires,
- le chiffrement asymétrique probabiliste,
- le chiffrement symétriques probabiliste, et
- le chiffrement symétrique déterministe surjectif (ou cipher) en présence de clés faibles.

Cette dernière primitive permet en particulier de modéliser le chiffrement par les mots de passe.

Les deux premiers types de chiffrement probabiliste sont inappropriés en présence de clés faibles, car ils procurent généralement un mécanisme de redondance permettant de détecter le succès du déchiffrement. Ce mécanisme a pour effet de divulguer immédiatement toute clé faible utilisée dans un chiffrement visible par l'attaquant.

A contrario, nous verrons que le chiffrement d'un nombre aléatoire frais — ou d'une donnée indistinguable d'un nombre aléatoire frais — par une clé faible w ne divulgue aucune information sur w lorsque le chiffrement utilisé est du troisième type, c'est-à-dire déterministe surjectif, pourvu que les nombres aléatoires suivent une distribution uniforme. (Pour le cas général, nous proposons également une condition technique suffisante.)

6.3.1 Sortes, termes et théorie équationnelle

L'ensemble de sortes τ que nous considérons dans cette section est défini par la grammaire suivante :

τ	::=		
	$\mid$	$SKey$	clés symétriques
	$\mid$	$EKey$	clés (publiques) de chiffrement
	$\mid$	$DKey$	clés (privées) de déchiffrement
	$\mid$	$Data$	mots de passe et autres données brutes
	$\mid$	$Coins$	jetons probabilistes pour le chiffrement
	$\mid$	$Pair[\tau_1, \tau_2]$	paires de messages
	$\mid$	$SCipher[\tau]$	chiffrements symétriques de message de type τ
	$\mid$	$ACipher[\tau]$	chiffrements asymétriques de message de type τ

L'ensemble des termes, dont les éléments sont notés $S, T, U, V, \ldots$, est construit sur un ensemble infini de variables x, y, ... et de noms a, b, n, r, k, sk, pk, ... pour chaque sorte, et sur les symboles de fonctions suivants :

enc_τ	:	$\tau \times Data \to \tau$	chiffrement déterministe
dec_τ	:	$\tau \times Data \to \tau$	déchiffrement associé
penc_τ	:	$\tau \times EKey \times Coins \to ACipher[\tau]$	chiffrement asymétrique
pdec_τ	:	$ACipher[\tau] \times DKey \to \tau$	déchiffrement asymétrique
pub	:	$DKey \to EKey$	extraction de clé publique
$\mathsf{pdec\_success}_\tau$	:	$ACipher[\tau] \times DKey \to Data$	prédicat pour le succès du déchiffrement asymétrique
senc_τ	:	$\tau \times SKey \times Coins \to SCipher[\tau]$	chiffrement symétrique
sdec_τ	:	$SCipher[\tau] \times SKey \to \tau$	déchiffrement symétrique
$\mathsf{sdec\_success}_\tau$	:	$SCipher[\tau] \times SKey \to Data$	prédicat pour le succès du déchiffrement symétrique
$\mathsf{pair}_{\tau_1,\tau_2}$	:	$\tau_1 \times \tau_2 \to Pair[\tau_1, \tau_2]$	appariement
$\mathsf{fst}_{\tau_1,\tau_2}$	:	$Pair[\tau_1, \tau_2] \to \tau_1$	première projection
$\mathsf{snd}_{\tau_1,\tau_2}$	:	$Pair[\tau_1, \tau_2] \to \tau_2$	seconde projection
$0, 1$	:	$Data$	constantes booléennes
$\mathsf{w}, \mathsf{c}_0, \mathsf{c}_1 \ldots$	:	$Data$	données arbitraires

En fonction de l'implémentation, nous permettons de n'inclure dans la signature que certains symboles de chiffrement et de déchiffrement. On note ainsi T_{penc} l'ensemble des types τ pour lesquels les symboles penc_τ, pdec_τ, and $\mathsf{pdec\_success}_\tau$ sont disponibles. De manière analogue, on définit les ensembles de types T_{senc} et T_{enc}. Dans la suite, nous supposerons que les paires ne sont pas chiffrées par des données brutes, c'est-à-dire que $T_{\mathsf{enc}} \cap \{Pair[\tau_1, \tau_2]\}_{\tau_1,\tau_2} = \emptyset$. Naturellement, il est toujours possible de chiffrer les paires composant par composant (de manière explicite).

Nos symboles de fonction ont pour but de modéliser différentes fonctions de chiffrement et de déchiffrement ainsi que des opérations auxiliaires. Les deux premières fonctions (enc_τ et dec_τ) sont destinées à recevoir des données brutes comme clés, notamment les symboles constants de la signature qui peuvent représenter les mots de passe d'un certain dictionnaire. (À l'inverse, les noms frais représentent des clés fortes.) Le choix d'un chiffrement déterministe enc_τ (sans paramètre de type *Coins*) est lié aux vulnérabilités posées par le chiffrement probabiliste en présence de clés faibles [AW05] : en effet, en cas de chiffrement probabiliste par un mot de passe, il est possible de déduire le mot de passe à partir de deux chiffrements d'un même message.

Notons que le langage ne donne aucun moyen à un attaquant de vérifier qu'une certaine valeur est bien un chiffré par enc_τ pour une clé donnée. Un telle propriété est essentielle pour prévenir les attaques par dictionnaire en pratique (par exemple dans le protocole EKE [BM92]). Les fonctions restantes sont plus habituelles, incluant les chiffrements asymétriques et symétriques (penc_τ et senc_τ) probabilistes au sens où ils admettent un ensemble de jetons de type *Coins* en paramètre.

Dans la suite, nous omettons la plupart du temps les annotations de type sur les symboles de fonction. Par exemple, étant donné S, T et U de type *Data*, nous écrivons $\mathsf{pair}(\mathsf{enc}(S,T),U)$ au lieu de $\mathsf{pair}_{Data,Data}(\mathsf{enc}_{Data}(S,T),U)$. De plus, nous utilisons au besoin les abréviations suivantes : $\{S\}_T$ pour $\mathsf{enc}(S,T)$, $\{S\}^r_{\mathsf{pub}(sk)}$ pour $\mathsf{penc}(S,\mathsf{pub}(sk),r)$, et $\{S\}^r_k$ pour $\mathsf{senc}(S,k,r)$.

Comme précédemment, la sémantique des primitives cryptographiques est modélisée par une théorie équationnelle. Nous considérons la théorie E engendrée par les équations suivantes :

$$\begin{aligned}
\mathsf{dec}_\tau(\mathsf{enc}_\tau(x,y),y) &\doteq x\\
\mathsf{enc}_\tau(\mathsf{dec}_\tau(x,y),y) &\doteq x\\
\mathsf{pdec}_\tau(\mathsf{penc}_\tau(x,\mathsf{pub}(y),z),y) &\doteq x\\
\mathsf{pdec_success}_\tau(\mathsf{penc}_\tau(x,\mathsf{pub}(y),z),y) &\doteq 1\\
\mathsf{sdec}_\tau(\mathsf{senc}_\tau(x,y,z),y) &\doteq x\\
\mathsf{sdec_success}_\tau(\mathsf{senc}_\tau(x,y,z),y) &\doteq 1\\
\mathsf{fst}_{\tau_1,\tau_2}(\mathsf{pair}_{\tau_1,\tau_2}(x,y)) &\doteq x\\
\mathsf{snd}_{\tau_1,\tau_2}(\mathsf{pair}_{\tau_1,\tau_2}(x,y)) &\doteq y\\
\mathsf{pair}_{\tau_1,\tau_2}(\mathsf{fst}_{\tau_1,\tau_2}(x),\mathsf{snd}_{\tau_1,\tau_2}(x)) &\doteq x
\end{aligned}$$

où x, y et z sont des variables de sorte appropriée.

L'équation $\mathsf{enc}_\tau(\mathsf{dec}_\tau(x,y),y) = x$ est spécifique au chiffrement par données faibles. Sans cette équation, un attaquant observant x est capable de deviner si x est bien un chiffré par la clé y de la manière simple suivante : déchiffrer x avec y, rechiffrer le résultat avec y, et comparer le résultat final avec x. L'équation implique que cette comparaison réussit toujours, et plus précisément que les expressions $\mathsf{enc}_\tau(n,\mathsf{c}_0)$ et $\mathsf{enc}_\tau(n,\mathsf{c}_1)$ sont indistinguables lorsque n est un nom frais de sorte τ. De telles propriétés sont importantes pour assurer la sécurité des protocoles comportant des secrets à faible entropie. En pratique, de nombreuses implémentations du chiffrement par mot de passe satisfont cette équation, notamment celles à base de permutations à clé (*keyed permutations*).

Nous notons $\mathcal{R}$ le système de réécriture correspondant aux équations ci-dessus, orientées de gauche à droite.

La théorie équationnelle précédente induit comme précédemment une notion d'équivalence statique entre environnements $\approx_E$. Nous terminons cette section par quelques exemples d'équivalences et de non-équivalences statiques pour la théorie E :

$$\{x \mapsto \{0\}_k^r\} \approx_E \{x \mapsto \{1\}_k^r\} \tag{6.3.1}$$

$$\{x \mapsto \{0\}_k^r,\ y \mapsto \{0\}_k^{r'}\} \approx_E \{x \mapsto \{1\}_k^r,\ y \mapsto \{0\}_{k'}^{r'}\} \tag{6.3.2}$$

$$\{x \mapsto \{n\}_{\mathsf{w}},\ y \mapsto \{m\}_{\mathsf{w}}\} \approx_E \{x \mapsto a_1,\ y \mapsto a_2\} \tag{6.3.3}$$

$$\{x \mapsto \{\{n\}_{\mathsf{w}}\}_{\mathsf{w}},\ y \mapsto \{m\}_{\mathsf{w}}\} \approx_E \{x \mapsto a_1,\ y \mapsto a_2\} \tag{6.3.4}$$

$$\{x \mapsto \{\{0\}^{r_1}_{\mathsf{pub}(sk)}\}_{\mathsf{w}},\ y \mapsto \{0\}^{r_2}_{\mathsf{pub}(sk)}\} \approx_E \{x \mapsto a_1,\ y \mapsto a_2\} \tag{6.3.5}$$

$$\{x \mapsto \{\{0\}^{r_1}_{\mathsf{pub}(sk)}\}_{\mathsf{w}},\ y \mapsto \{0\}^{r_1}_{\mathsf{pub}(sk)}\} \approx_E \{x \mapsto \{a_1\}_{\mathsf{w}},\ y \mapsto a_1\} \tag{6.3.6}$$

$$\{x \mapsto \{\{n\}_k^{r_1}\}_{\mathsf{w}},\ y \mapsto k\} \not\approx_E \{x \mapsto a_1,\ y \mapsto k\} \tag{6.3.7}$$

Les exemples (6.3.1) et (6.3.2) montrent que le chiffrement symétrique par des clés fortes masque toute information sur les messages chiffrés — y compris l'égalité — et sur les clés utilisées. Les exemples (6.3.3) et (6.3.4) illustrent le fait que le chiffrement des noms frais sous une constante w (intuitivement, un secret faible) se comportent comme des noms frais.

L'exemple (6.3.5) est similaire à (6.3.4) : il illustre le fait qu'un chiffré par une clé publique $\{0\}^{r_1}_{\mathsf{pub}(sk)}$ se comporte encore comme un nombre aléatoire une fois chiffré par w. Dans les exemples (6.3.3)–(6.3.5), les messages en clair ne sont pas disponibles d'une autre manière à l'observateur. L'exemple (6.3.6) traite du cas où un observateur obtient également le message chiffré par y ; dans ce cas, il peut observer la relation entre les valeurs de x et de y (en l'occurrence $y \doteq \{x\}_w$). L'exemple (6.3.7) indique qu'un observateur ayant accès à k peut distinguer $\{\{n\}_k^{r_1}\}_{\mathsf{w}}$ d'un nombre aléatoire : en effet, une fois cette expression déchiffrée avec w, l'adversaire peut ensuite vérifier si le résultat est un chiffré valide par k ou non.

6.3.2 Implémentation

Nous décrivons maintenant l'interprétation concrète des symboles de fonction, et les hypothèses cryptographiques qui s'y rapportent. Comme précédemment, on note η un certain paramètre de complexité.

Schémas de chiffrement. Dans la suite, nous fixons un schéma de chiffrement à clé publique $\Pi^p = (\mathcal{K}^p, \mathcal{E}^p, \mathcal{D}^p)$ et à clé symétrique $\Pi^s = (\mathcal{K}^s, \mathcal{E}^s, \mathcal{D}^s)$. Nous fixons également un schéma de chiffrement symétrique, déterministe et préservant les types, noté $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Par *préservant les types*, on entend que, pour tout type $\tau \in T_{\mathsf{enc}}$, le chiffrement et le déchiffrement par Π envoient l'interprétation du type τ, notée $[\![\tau]\!]_\eta$, sur elle-même. Dans chacun de ces triplets, la première composante est l'algorithme de génération de clés, la deuxième la fonction de chiffrement, et la dernière la fonction de déchiffrement.

Dans le cas du chiffrement à clé publique Π^p, quitte à augmenter la taille des clés privées, on suppose qu'il existe une certaine fonction d'extraction permettant d'associer (en temps polynomial) chaque clé privée à sa clé publique.

Pour chaque η, nous écrivons $k \xleftarrow{R} \mathcal{K}_\eta$ et $k \xleftarrow{R} \mathcal{K}^s_\eta$ les processus de génération d'une clé de chiffrement k pour Π et Π^s, respectivement. De manière similaire, $(pk, sk) \xleftarrow{R} \mathcal{K}^p_\eta$ représente la génération d'une paire (pk, sk) de clés de chiffrement et de déchiffrement pour Π^p.

Les fonctions de chiffrement $\mathcal{E}^p$ et $\mathcal{E}^s$ sont probabilistes comme habituellement. Nous notons $\mathcal{E}^p(m, k, r)$ et $\mathcal{E}^s(m, k, r)$ les chiffrements respectifs d'un message m par un clé k avec les jetons probabilistes r. Nous notons également $c \xleftarrow{R} \mathcal{E}^p(m, k)$ et $c \xleftarrow{R} \mathcal{E}^s(m, k)$ les processus de chiffrement correspondants, utilisant des jetons probabilistes frais.

Enfin, on suppose que l'ensemble des clés pour Π est de la forme $\{0,1\}^{\alpha_1(\eta)}$, et que l'ensemble des jetons pour Π^s et Π^p est $\{0,1\}^{\alpha_2(\eta)}$, où les fonctions $\alpha_1(\eta)$ et $\alpha_2(\eta)$ sont bornées polynomialement et au moins linéairement croissantes.

Sortes, primitives et tirages aléatoires. Nous définissons l'interprétation $[\![\tau]\!]_\eta$ d'un type τ de manière inductive :

$$\begin{aligned}
[\![SKey]\!]_\eta &= \text{"}SKey\text{"} \,\|\, \{\text{clés symétriques pour } \Pi^s(\eta)\} \\
[\![EKey]\!]_\eta &= \text{"}EKey\text{"} \,\|\, \{\text{clés publiques pour } \Pi^p(\eta)\} \\
[\![DKey]\!]_\eta &= \text{"}DKey\text{"} \,\|\, \{\text{clés privées pour } \Pi^p(\eta)\} \\
[\![Data]\!]_\eta &= \text{"}Data\text{"} \,\|\, \{0,1\}^{\alpha_1(\eta)} \\
[\![Coins]\!]_\eta &= \text{"}Coins\text{"} \,\|\, \{0,1\}^{\alpha_2(\eta)} \\
[\![Pair[\tau_1, \tau_2]]\!]_\eta &= \text{"}Pair\text{"} \,\|\, [\![\tau_1]\!]_\eta \,\|\, [\![\tau_2]\!]_\eta \\
[\![SCipher[\tau]]\!]_\eta &= \text{"}SCipher\text{"} \,\|\, \tau \,\|\, \{\text{chiffrés par } \Pi^s(\eta)\} \\
[\![ACipher[\tau]]\!]_\eta &= \text{"}ACipher\text{"} \,\|\, \tau \,\|\, \{\text{chiffrés par } \Pi^p(\eta)\}
\end{aligned}$$

où $\|$ dénote la concaténation des chaînes de bits (et par extension celle d'ensembles de chaînes de bits), et nous supposons fixé un encodage de chaque expression de type τ dans une chaîne de bits. Par souci de simplification, nous supposons que chaque type τ ci-dessus correspond à des données $[\![\tau]\!]_\eta$ de taille uniforme.

L'implémentation des symboles de fonction est alors définie de la manière suivante :

- Les symboles $\mathsf{pair}_{\tau_1,\tau_2}$, $\mathsf{fst}_{\tau_1,\tau_2}$ et $\mathsf{snd}_{\tau_1,\tau_2}$ sont implémentés par la concaténation et les projections correspondantes sur les chaînes de bits, de la manière habituelle.
- Les constantes w, c_0, c_1, ... sont envoyés vers des familles arbitraires de chaînes de bits, calculables en temps polynomial (probabiliste) et de taille $\alpha_1(\eta)$, auxquelles on ajoute le préfixe "$Data$" ; 0 et 1 sont interprétés respectivement par "$Data$" $\| 0^{\alpha_1(\eta)}$ et "$Data$" $\| 1^{\alpha_1(\eta)}$.
- Pour chaque $\tau \in T_{\mathsf{penc}}$, les implémentations de penc_τ, pdec_τ, et $\mathsf{pdec\_success}_\tau$ sont définies comme suit :

$$[\![\mathsf{penc}_\tau]\!]_\eta(m, \text{"}EKey\text{"}||pk, \text{"}Coins\text{"}||r) = \text{"}ACipher\text{"}\|\tau\|\mathcal{E}^p_\tau(m, pk, r)$$

$$[\![\mathsf{pdec}_\tau]\!]_\eta(m, \text{"}DKey\text{"}||sk) = \begin{cases} \mathcal{D}^p(c, sk) & \text{si } m = \text{"}ACipher\text{"}\|\tau\|c \text{ et le} \\ & \text{déchiffrement } \mathcal{D}^p(c, sk) \text{ réussit} \\ \langle \text{une valeur} & \\ \text{quelconque} \rangle & \text{sinon} \end{cases}$$

$$[\![\mathsf{pdec\_success}_\tau]\!]_\eta(m, "DKey"||sk) = "Data" \;\|$$
$$\begin{cases} 1^{\alpha_1(\eta)} & \text{si } m = "ACipher"\|\tau\|c \text{ et le déchiffrement } \mathcal{D}^p(c, sk) \text{ réussit} \\ 0^{\alpha_1(\eta)} & \text{sinon} \end{cases}$$

Les implémentations de senc_τ, sdec_τ et $\mathsf{sdec\_success}_\tau$ pour chaque $\tau \in T_{\mathsf{senc}}$ sont définies de manière similaire.

– Pour chaque $\tau \in T_{\mathsf{enc}}$, les implémentations de enc_τ et de dec_τ sont définies par

$$\begin{array}{rcl} [\![\mathsf{enc}_\tau]\!]_\eta(m, "Data"\|k) & = & \mathcal{E}(m,k) \\ [\![\mathsf{dec}_\tau]\!]_\eta(c, "Data"\|k) & = & \mathcal{D}(c,k) \end{array}$$

Nous supposons que $\mathcal{E}(\cdot,k)$ et $\mathcal{D}(\cdot,k)$ sont des bijections réciproques de l'ensemble $[\![\tau]\!]_\eta$ sur lui-même. En particulier, les étiquettes (tags) sont conservées par ces fonctions.

– Enfin, le symbole pub est implémenté par la fonction d'extraction des clés publiques mentionnée plus haut.

Le tirage des valeurs aléatoires de type τ ($e \xleftarrow{R} [\![\tau]\!]_\eta$) est défini par induction sur τ (en ajoutant les étiquettes appropriées dans chaque cas) :

– Si τ est égale à *SKey*, *EKey* ou *DKey*, utiliser l'algorithme de génération de clés dédié : respectivement $\mathcal{K}^s$, $\mathsf{fst}(\mathcal{K}^p)$ et $\mathsf{snd}(\mathcal{K}^p)$.
– Si τ vaut *Data* ou *Coins*, utiliser la distribution uniforme sur $[\![\tau]\!]_\eta$.
– Si $\tau = Pair[\tau_1, \tau_2]$, tirer récursivement deux éléments aléatoires à partir de $[\![\tau_1]\!]_\eta$ et $[\![\tau_2]\!]_\eta$, puis concaténer et étiqueter les valeurs obtenues.
– Si τ vaut $SCipher[\tau]$ ou $ACipher[\tau]$, chiffrer un élément de $[\![\tau]\!]_\eta$ par une clé fraîche aléatoire de la sorte appropriée.

Hypothèses cryptographiques. Nous décrivons maintenant les hypothèses requises sur l'implémentation en vue de justifier la relation d'équivalence statique.

En ce qui concerne les chiffrements symétrique et asymétrique, nous considérons des schémas satisfaisant une notion de sécurité proche des notions de sécurité de *type 0* et de *type 1* [AR02]. Intuitivement, nous demandons que pour chaque sorte τ, la fonction de chiffrement restreinte aux éléments de $[\![\tau]\!]$ ne révèle aucune information sur la clé utilisée pour le chiffrement et masque toute information partielle sur les messages en clair — excepté leur appartenance à l'ensemble $[\![\tau]\!]$.

Définition 6.1. Soit $\Pi^s = (\mathcal{K}^s, \mathcal{E}^s, \mathcal{D}^s)$ un schéma de chiffrement symétrique. Pour chaque paramètre de sécurité η et type $\tau \in T_{\mathsf{senc}}$, on considère l'expérience suivante mettant en jeu un adversaire PPTIME en deux étapes $A = (A_1, A_2)$:

– une clé fraîche k est générée via $k \xleftarrow{R} \mathcal{K}^s(\eta)$;
– A_1 interagit avec l'oracle $\mathcal{E}^s(\cdot,k)$, c'est-à-dire que A_1 peut soumettre des messages m à l'oracle et obtenir en réponse les chiffrés correspondants $\mathcal{E}^s(m,k)$;
– ensuite A_1 envoie un message $m^* \in [\![\tau]\!]_\eta$ en guise de « défi » ainsi qu'une donnée st ;
– un bit $b \xleftarrow{R} \{0,1\}$ est choisi au hasard ; si $b = 0$, soit c le chiffrement (étiqueté) de m^* par k, c'est-à-dire $c \xleftarrow{R} "SCipher"\|\tau\|\mathcal{E}^s(m^*,k)$; dans le cas contraire, soit c le chiffrement (étiqueté) d'un élément aléatoire de type τ par une clé aléatoire : $c \xleftarrow{R} [\![SCipher[\tau]]\!]_\eta$;
– A_2 reçoit c et st, et répond par un bit b'.

On dit que l'adversaire A *gagne le jeu* ssi $b' = b$. L'*avantage* de A est défini par

$$\mathrm{Adv}^{\tau}_{\Pi^s,A}(\eta) = \mathbb{P}\left[A \text{ gagne}\right] - \frac{1}{2}.$$

Enfin, un schéma de chiffrement Π^s est dit T_{senc}-*sûr* si pour tout adversaire PPTIME A et tout $\tau \in T_{\mathsf{senc}}$, la fonction $\mathrm{Adv}^{\tau}_{\Pi^s,A}(\cdot)$ est négligeable.

Définition 6.2. Soit $\Pi^p = (\mathcal{K}^p, \mathcal{E}^p, \mathcal{D}^p)$ un schéma de chiffrement asymétrique. Pour chaque paramètre de complexité η et type $\tau \in T_{\mathsf{penc}}$, on considère l'expérience suivante, impliquant un adversaire PPTIME en deux étapes $A = (A_1, A_2)$:

- une paire de clés de chiffrement et de déchiffrement (pk, sk) est générée par $(pk, sk) \xleftarrow{R} \mathcal{K}^p(\eta)$, et A_1 reçoit pk ;
- A_1 renvoie un message $m^* \in [\![\tau]\!]_\eta$ ainsi qu'une donnée st ;
- un bit $b \xleftarrow{R} \{0,1\}$ est choisi aléatoirement ; si $b = 0$, soit c le chiffrement (étiqueté) de m^* par pk, c'est-à-dire $c \xleftarrow{R}$ "*ACipher*"$\|\tau\|\mathcal{E}^p(m^*, pk)$; dans le cas contraire, soit c le chiffrement (étiqueté) d'un élément aléatoire de sorte τ par une clé publique aléatoire, autrement dit $c \xleftarrow{R} [\![\mathit{ACipher}[\tau]]\!]_\eta$;
- A_2 reçoit c et st, et répond par un bit b'.

On dit que l'adversaire A *gagne le jeu* ssi $b' = b$. L'*avantage* de A est défini par

$$\mathrm{Adv}^{\tau}_{\Pi^p,A}(\eta) = \mathbb{P}\left[A \text{ gagne}\right] - \frac{1}{2}.$$

Enfin, un schéma de chiffrement Π^p est dit T_{penc}-*sûr* si pour tout adversaire PPTIME A et tout $\tau \in T_{\mathsf{penc}}$, la fonction $\mathrm{Adv}^{\tau}_{\Pi^p,A}(\cdot)$ est négligeable.

Notre notion de sécurité pour le chiffrement déterministe est la conjonction d'une condition standard de chiffrement et d'une condition moins standard, relative au chiffrement par des clés faibles.

Définition 6.3. Soit $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ un schéma de chiffrement symétrique, déterministe, préservant les types, ayant pour espace de clés $\{0,1\}^{\alpha_1(\eta)}$ pour chaque η.

1. **Sécurité « Real-or-Random » (T_{enc}-RoR) :** Pour chaque paramètre de sécurité η et type $\tau \in T_{\mathsf{enc}}$, on considère l'expérience suivante, impliquant un adversaire PPTIME en deux étapes $A = (A_1, A_2)$:
   - une clé k est générée via $k \xleftarrow{R} \mathcal{K}(\eta)$;
   - A_1 interagit avec l'oracle $\mathcal{E}(\cdot, k)$; autrement dit, A_1 peut envoyer des messages (étiquetés) m à l'oracle et recevoir en retour les chiffrés (étiquetés) $\mathcal{E}(m, k)$;
   - ensuite A_1 retourne un message $m^* \in [\![\tau]\!]_\eta$ et une information d'état st ;
   - un bit $b \xleftarrow{R} \{0,1\}$ est tiré au sort ; si $b = 0$, soit c la chiffrement (étiqueté) de m^* par k, c'est-à-dire, $c = \mathcal{E}(m^*, k)$; sinon, $c \xleftarrow{R} [\![\tau]\!]_\eta$ prend une valeur aléatoire dans $[\![\tau]\!]_\eta$;
   - A_2 reçoit c et st, et retourne un bit b'.

   L'adversaire A gagne si $b' = b$ *et* le message m^* est distinct de tous les messages m soumis par A à l'oracle de chiffrement. L'avantage de A est par définition $\mathrm{Adv}^{\tau}_{\mathrm{RoR},\Pi,A}(\eta) = \mathbb{P}[A \text{ gagne}] - \frac{1}{2}$. On dit que Π est T_{enc}-*RoR-sûr* si pour tout adversaire PPTIME A et tout $\tau \in T_{\mathsf{enc}}$, la fonction $\mathrm{Adv}^{\tau}_{\mathrm{RoR},\Pi,A}(\cdot)$ est négligeable.

2. **Chiffrement par données faibles (T_{enc}-Pwd) :** Pour chaque valeur du paramètre de sécurité η et type $\tau \in T_{\mathsf{enc}}$, on considère l'expérience suivante, impliquant un adversaire PPTIME en deux étapes $A = (A_1, A_2)$:
   - A_1 émet une clé $k \in \{0,1\}^{\alpha_1(\eta)}$ et une information d'état st ;
   - un bit $b \xleftarrow{R} \{0,1\}$ est tiré au sort ; si $b = 0$, soit c le chiffrement (étiqueté) d'un nombre aléatoire par k, c'est-à-dire, $m \xleftarrow{R} [\![\tau]\!]_\eta$ et $c = \mathcal{E}(m,k)$; sinon, soit $c \xleftarrow{R} [\![\tau]\!]_\eta$ un élément tiré au sort dans $[\![\tau]\!]_\eta$;
   - A_2 reçoit c et st, et répond un bit b'.

   L'adversaire A gagne si $b' = b$. L'adversaire A gagne si $b' = b$. L'avantage de A est par définition $\mathrm{Adv}^{\tau}_{\mathrm{Pwd},\Pi,A}(\eta) = \mathbb{P}[A \text{ gagne}] - \frac{1}{2}$. On dit que Π est T_{enc}-*Pwd-sûr* si pour tout adversaire PPTIME A et tout $\tau \in T_{\mathsf{enc}}$, la fonction $\mathrm{Adv}^{\tau}_{\mathrm{Pwd},\Pi,A}(\cdot)$ est négligeable.

Pour finir, Π est T_{enc}-*sûr* s'il est à la fois T_{enc}-RoR et T_{enc}-Pwd sûr.

La condition 1 (sûreté T_{enc}-RoR) est une variante du critère de sécurité IND-P1-C0 [PP04, BCK05]. Cette propriété est requise du fait que nous autorisons l'utilisation de enc en tant qu'algorithme de chiffrement à part entière, c'est-à-dire également avec des clés fortes. Sans cette condition, le résultat de cette section reste valable si l'on restreint les clés de enc aux constantes — comme c'est le cas dans le formalisme de M. Abadi et de B. Warinschi [AW05].

La condition 2 (sûreté T_{enc}-Pwd) concerne la sécurité des mots de passe (et autres données faibles) quand ils sont utilisés comme clés. Intuitivement, cette condition contraint le chiffrement d'une donnée aléatoire à être distribué, au moins en apparence, comme la valeur elle-même. Une condition similaire introduite par M. Abadi et B. Warinschi [AW05] donne la possibilité aux distributions des valeurs et des chiffrements d'être deux distributions (fixées) différentes. La différence ici est due au fait que nous autorisons plusieurs couches de chiffrement par des mots de passe (voir l'exemple (6.3.4)).

Notons que la condition 2 découle directement des propriétés algébriques d'une permutation à clé, lorsque les distributions $[\![\tau]\!]$ (étiquette exclue) sont uniformes, ou simplement indistinguables de distributions uniformes.

Finalement, une implémentation basée sur les schémas (Π^s, Π^p, Π) sera dite $(T_{\mathsf{senc}}, T_{\mathsf{penc}}, T_{\mathsf{enc}})$-*sûre* (ou simplement *sûre*) si les trois schémas Π^s, Π^p et Π sont respectivement T_{senc}-sûr, T_{penc}-sûr et T_{enc}-sûr.

La construction d'une implémentation sûre dans le modèle cryptographique standard est détaillée dans la version longue de l'article original [ABW06].

6.3.3 Correction cryptographique de l'équivalence statique

Nous énonçons maintenant le résultat principal de cette section. Dans la lignée de [AR02], il nous faut introduire un certain nombre d'hypothèses de bonne formation sur les environnements considérés, excluant notamment les cycles de chiffrement.

Une *position de clé* dans un terme ou un environnement est une position correspondant à l'argument U d'un sous-terme de la forme $\mathsf{pub}(U)$, ou au second argument V d'un sous-terme $\mathsf{enc}_\tau(U,V)$, $\mathsf{penc}_\tau(U,V,W)$, ou $\mathsf{senc}_\tau(U,V,W)$.

Un *cycle de chiffrement* dans un environnement φ est une suite de noms k_0, k_1, …, k_n de sortes *Data*, *DKey*, ou *SKey* tels que $k_n = k_0$ et

pour tout $0 \leq i \leq n-1$, il existe un sous-terme de φ de la forme $\mathsf{enc}_\tau(U,V)$, $\mathsf{penc}_\tau(U,V,W)$, ou $\mathsf{senc}_\tau(U,V,W)$ tel que k_i est un sous-terme de U *non* en position de clé, et k_{i+1} est un sous-terme de V.

Par exemple, l'environnement $\varphi_1 = \{x \mapsto \{sk_1\}^{r_1}_{k_2},\ y \mapsto \{k_2\}^{r_2}_{\mathsf{pub}(sk_1)}\}$ admet un cycle de chiffrement, tandis que $\varphi_2 = \{x \mapsto \{\mathsf{pub}(sk_1)\}^{r_1}_{k_2},\ y \mapsto \{k_2\}^{r_2}_{\mathsf{pub}(sk_1)}\}$ en est exempt.

Un environnement φ est *bien formé* s'il satisfait les conditions suivantes :

(i) les termes de φ sont en forme $\mathcal{R}$-normale ;

(ii) φ ne contient pas de symboles dec, pdec, sdec, pdec_success, sdec_success, fst, et snd ;

(iii) les termes en position de clé dans φ sont de la forme suivante, selon leur sorte :
- sortes *DKey* et *SKey* : noms,
- sorte *EKey* : noms et termes de la forme $\mathsf{pub}(a)$,
- sorte *Data* : noms et constantes ;

(iv) les termes de sorte *Coins* dans φ sont uniquement des noms, et apparaissent comme troisième argument d'un symbole de chiffrement ; de plus, si un tel nom apparaît à deux endroits dans φ, alors les autres arguments du chiffrement (message chiffré et clé) sont identiques ;

(v) φ n'a pas de cycle de chiffrement ;

(vi) pour tout sous-terme de φ de la forme $\mathsf{enc}(T,k)$ où k est un nom, T ne contient aucunes constantes w, c_0, c_1, ..., et T n'a pas de sous-terme de la forme $\mathsf{enc}(S,0)$ ou $\mathsf{enc}(S,1)$.

La condition (ii) correspond au fait que nous nous intéressons à l'indistinguabilité des expressions sans destructeurs. La condition (iii) signifie que les clés de chiffrements doivent être purement aléatoires, excepté pour le chiffrement déterministe où nous autorisons les constantes (modélisant par exemple une clé faible). De la même manière, la condition (iv) signifie que les jetons probabilistes sont des aléas purs, frais pour chaque nouveau chiffrement. La condition (v) interdit les cycles de chiffrement. Enfin, la condition (vi) restreint les occurrences des constantes à l'intérieur des messages chiffrés par le chiffrement déterministe pour des clés fortes. Par exemple, la condition exclut l'environnement $\{x_1 \mapsto \mathsf{enc}(\mathsf{c}_1,k),\ x_2 \mapsto \mathsf{enc}(\mathsf{c}_2,k)\}$, équivalent à $\{x_1 \mapsto a_1,\ x_2 \mapsto a_2\}$ formellement mais non du point de vue cryptographique si c_1 et c_2 ont la même implémentation. Plus généralement, si T_1 et T_2 vérifient $T_1 \neq_E T_2$, les chiffrements $\mathsf{enc}(T_1,k)$ et $\mathsf{enc}(T_2,k)$ se comportent formellement comme deux noms frais distincts. Concrètement, il se peut que ce ne soit pas le cas si les valeurs de T_1 et T_2 ont une chance non négligeable d'entrer en collision.

Nous pouvons maintenant énoncer le résultat escompté :

Théorème 6.5 ($\approx_E$-sûreté)**.** *Soit φ_1 et φ_2 deux environnements bien formés tels que $\varphi_1 \approx_E \varphi_2$. Pour toute implémentation sûre, on a $[\![\varphi_1]\!] \approx [\![\varphi_2]\!]$.*

La preuve de ce théorème fait l'objet de la section 6.4. Elle repose sur une procédure de décision originale de l'équivalence statique pour la théorie concernée, dont on montre la correction formelle et cryptographique de chacune des étapes.

Nous laissons ouverte la question de la complétude de l'équivalence statique sur les environnements bien formés, c'est-à-dire l'existence d'une réciproque au théorème 6.5. Notons toutefois que par la proposition 5.2 du chapitre 5, il suffit de montrer la $=_E$-fidélité de l'implémentation pour obtenir la $\approx_E$-fidélité, c'est-à-dire une forme renforcée de complétude.

6.3.4 Application aux attaques par dictionnaire

Comme nous l'avons vu au chapitre 3 (section 3.2), il est possible d'énoncer la résistance aux attaques par dictionnaire en termes d'équivalence observationnelle entre processus. Dans le cas de la sécurité des données statiques, c'est-à-dire pour un attaquant purement passif, l'équivalence observationnelle se réduit à l'équivalence statique entre environnements.

Soit w une constante modélisant une certaine donnée faible, par exemple un mot de passe. On dit que w est *caché dans l'environnement* φ si on a l'équivalence $\varphi\{\mathsf{w} \mapsto \mathsf{c}_0\} \approx_E \varphi\{\mathsf{w} \mapsto \mathsf{c}_1\}$. Les remplacements $\{\mathsf{w} \mapsto \mathsf{c}_0\}$ et $\{\mathsf{w} \mapsto \mathsf{c}_1\}$ correspondent aux instanciations d'un mot de passe par deux valeurs distinctes. On obtient la correction cryptographique de cette notion directement comme un corollaire du théorème 6.5, de même sa généralisation immédiate à des mots de passe multiples.

Corollaire 6.6 (Mot de passe simple)**.** *Supposons l'implémentation sûre. Soit φ un environnement bien formé, w une constante de sorte Data, et c_0, c_1 deux constantes fraîches distinctes de sorte Data. Si $\varphi\{\mathsf{w} \mapsto \mathsf{c}_0\} \approx_E \varphi\{\mathsf{w} \mapsto \mathsf{c}_1\}$ alors w est caché concrètement dans φ : pour toutes suites de chaînes de bits PPTIME κ_0, κ_1 telles que $\kappa_i(\eta) \in \{0,1\}^{\alpha_1(\eta)}$, on a*

$$[\![\varphi]\!]_{\eta,\mathsf{w}\mapsto\kappa_0(\eta)} \approx [\![\varphi]\!]_{\eta,\mathsf{w}\mapsto\kappa_1(\eta)}$$

Corollaire 6.7 (Mots de passe multiples)**.** *Supposons l'implémentation sûre. Soit φ un environnement bien formé, $\mathsf{w}_1, \ldots, \mathsf{w}_n$ n constantes de sorte Data, et c_0, c_1, $\ldots$, c_n $n+1$ constantes fraîches distinctes de sorte Data. Si $\varphi\{\mathsf{w}_1 \mapsto \mathsf{c}_1, \ldots, \mathsf{w}_n \mapsto c_n\} \approx_E \varphi\{\mathsf{w}_1 \mapsto \mathsf{c}_0, \ldots, \mathsf{w}_n \mapsto c_0\}$ alors $\mathsf{w}_1, \ldots, \mathsf{w}_n$ sont cachées concrètement dans φ : pour toutes suites de chaînes de bits PPTIME $\kappa_1, \ldots, \kappa_n, \kappa'_1, \ldots, \kappa'_n$ telles que $\kappa_i(\eta), \kappa'_i(\eta) \in \{0,1\}^{\alpha_1(\eta)}$, on a*

$$[\![\varphi]\!]_{\eta,\mathsf{w}_1\mapsto\kappa_1(\eta),\ldots,\mathsf{w}_n\mapsto\kappa_n(\eta)} \approx [\![\varphi]\!]_{\eta,\mathsf{w}_1\mapsto\kappa'_1(\eta),\ldots,\mathsf{w}_n\mapsto\kappa'_n(\eta)}$$

Exemple 6.1. À titre d'application, considérons le protocole Encrypted Key Exchange (EKE) [BM92, AW05]). Le but de ce protocole est l'établissement d'une clé de session authentifiée k_1 entre deux agents A et B ne partageant initialement qu'un mot de passe w_{AB}, potentiellement faible. Les messages échangés dans une session normale de ce protocole s'écrivent :

$$\begin{array}{lll} 0. & A \to B : & \{pk\}_{\mathsf{w}_{AB}} \\ 1. & B \to A : & \{\{k_1\}^{r_1}_{pk}\}_{\mathsf{w}_{AB}} \\ 2. & A \to B : & \{k_A\}^{r_2}_{k_1} \\ 3. & B \to A : & \{\mathsf{pair}(k_A,\, k_B)\}^{r_3}_{k_1} \\ 4. & A \to B : & \{k_B\}^{r_4}_{k_1} \end{array}$$

Ainsi, A génère initialement une paire de clés privée et publique fraîche (sk, pk), puis envoie la clé publique pk chiffrée par le mot de passe w_{AB}. Utilisant sa connaissance de w_{AB}, B récupère pk, génère une clé symétrique fraîche k_1 et la fait parvenir à A, chiffrée par pk et w_{AB}. A peut donc récupérer k_1 et entreprendre avec B un échange classique de messages de confirmation (k_A et k_B servent de défi ou « *challenge* »).

Dans le cas d'un intrus passif, les messages d'une session du protocole correspondent à l'environnement suivant :

$$\varphi = \{x_1 \mapsto \{pk\}_{\mathsf{w}_{AB}},\ x_2 \mapsto \{\{k_1\}^{r_1}_{pk}\}_{\mathsf{w}_{AB}},\ x_3 \mapsto \{k_A\}^{r_2}_{k_1},\\ x_4 \mapsto \{\mathsf{pair}(k_A,\, k_B)\}^{r_3}_{k_1},\ x_5 \mapsto \{k_B\}^{r_4}_{k_1}\ \}$$

où selon les conventions précédentes $pk, k_1, k_A, k_B, r_1, \ldots, r_4$ sont des noms (toujours privés) de sorte appropriée. Soit $\varphi_i = \varphi\{\mathsf{w}_{AB} \mapsto \mathsf{c}_i\}$ pour $i \in \{0,1\}$.

Nous vérifions plus loin que $\varphi_0 \approx_E \varphi_1$, de sorte que le corollaire 6.6 s'applique. Pour une implémentation sûre au sens précédent, le mot de passe w_{AB} est bien caché concrètement dans φ.

Notons que grâce au corollaire 6.7, nous pouvons traiter de la même manière un nombre fixé, quelconque de sessions.

Exemple 6.2. Inversement, considérons la variante suivante du protocole EKE, où le second message utilise maintenant du chiffrement symétrique.

$$\begin{array}{lll} 0. & A \to B : & \{k_0\}_{\mathsf{w}_{AB}} \\ 1. & B \to A : & \{\{k_1\}^{r_1}_{k_0}\}_{\mathsf{w}_{AB}} \\ 2. & A \to B : & \{k_A\}^{r_2}_{k_1} \\ 3. & B \to A : & \{\mathsf{pair}(k_A,\, k_B)\}^{r_3}_{k_1} \\ 4. & A \to B : & \{k_B\}^{r_4}_{k_1} \end{array}$$

L'environnement correspondant à une session principale s'écrit :

$$\varphi = \{x_1 \mapsto \{k_0\}_{\mathsf{w}_{AB}},\ x_2 \mapsto \{\{k_1\}^{r_1}_{k_0}\}_{\mathsf{w}_{AB}},\ x_3 \mapsto \{k_A\}^{r_2}_{k_1},\\ x_4 \mapsto \{\mathsf{pair}(k_A,\, k_B)\}^{r_3}_{k_1},\ x_5 \mapsto \{k_B\}^{r_4}_{k_1}\ \}$$

Soit $\varphi_i = \varphi\{\mathsf{w}_{AB} \mapsto \mathsf{c}_i\}$ pour $i \in \{0,1\}$.

Cette fois-ci, $\varphi_0 \not\approx_E \varphi_1$. Soit en effet $M = \mathsf{sdec\_success}(\mathsf{dec}(x_2, \mathsf{c}_0),\, \mathsf{dec}(x_1, \mathsf{c}_0))$. On a $M\varphi_0 =_E 1$ mais $M\varphi_0 \neq_E 1$.

Autrement dit, il est possible de récupérer le bon mot de passe par énumération exhaustive en testant si le déchiffrement du premier message est bien la clé symétrique k_0 utilisée dans le second.

6.4 Preuve du théorème 6.5

L'objet de cette section est prouver le théorème principal de la section précédente. Après quelques notations, nous décrivons un ensemble de règle de transformation utilisées pour caractériser l'équivalence statique entre environnements bien formés. Ces règles ont pour but de simplifier une *équation* $\varepsilon = (\varphi_1 \approx^? \varphi_2)$ sous une forme plus petite (dans un certain sens) : $\varepsilon' = (\varphi_1' \approx^? \varphi_2')$, $\varepsilon' = \top$, ou $\varepsilon' = \bot$.

Définition 6.4. Une équation ε est *vraie formellement* ssi $\varepsilon = \top$, ou bien $\varepsilon = (\varphi_1 \approx^? \varphi_2)$ et les deux environnements sont statiquement équivalents dans E, c'est-à-dire, $\varphi_1 \approx_E \varphi_2$.

L'équation ε est *vraie calculatoirement* ssi $\varepsilon = \top$, ou bien $\varepsilon = (\varphi_1 \approx^? \varphi_2)$ et les deux environnements sont indistinguables, c'est-à-dire, $[\![\varphi_1]\!] \approx [\![\varphi_2]\!]$.

Dans la suite, nous nous efforçons de définir une relation de transformation $\Longrightarrow$ vérifiant les propriétés suivantes :

- **(Correction formelle)** Si $\varepsilon \Longrightarrow \varepsilon'$, alors ε est vraie formellement ssi ε' l'est.
- **(Progression)** Si $\varepsilon \notin \{\top, \bot\}$, alors il existe ε' tel que $\varepsilon \Longrightarrow \varepsilon'$.
- **(Sûreté calculatoire)** Supposons l'implémentation sûre. Si $\varepsilon \Longrightarrow \varepsilon'$ et ε' est vraie calculatoirement, alors c'est le cas de ε.
- **(Terminaison)** Il n'existe pas de séquence infinie de transformations $\varepsilon_1 \Longrightarrow \varepsilon_2 \Longrightarrow \ldots \Longrightarrow \varepsilon_n \ldots$

L'ensemble de ces propriétés implique bien le résultat du théorème 6.5 : si $\varphi_1 \approx_E \varphi_2$ et les deux environnements φ_1 et φ_2 sont bien formés alors $[\![\varphi_1]\!] \approx [\![\varphi_2]\!]$. De plus, comme la correction formelle est énoncée dans les deux directions et que les règles de transformation sont effectives, on en déduit une procédure de décision pour l'équivalence statique entre environnements bien formés.

6.4.1 Notations

Nous commençons par introduire quelques notations utiles. Comme suggéré plus haut, une *équation* ε est une expression de la forme $\varphi_1 \approx^? \varphi_2$, $\top$ ou $\bot$.

Étant donné un terme U, n signes $\ell_1, \ldots, \ell_n \in \{-1, +1\}$ ($n \geq 0$), et n termes $V_1, \ldots, V_n$ de sorte *Data*, on définit le terme $\{U\}_{V_1^{\ell_1} \cdot \cdots \cdot V_n^{\ell_n}}$ récursivement par

$$\begin{aligned} \{U\}_{\epsilon} &= U \\ \{U\}_{W \cdot V^{+1}} &= \mathsf{enc}(\{U\}_W, V) \\ \{U\}_{W \cdot V^{-1}} &= \mathsf{dec}(\{U\}_W, V) \end{aligned}$$

où ϵ désigne le mot vide et où $\cdot$ est vu comme un symbole associatif. Par la suite on identifie V et V^{+1}. Si $\vec{V}$ désigne l'expression formelle $\vec{V} = V_1^{\ell_1} \cdot \cdots \cdot V_n^{\ell_n}$, on écrit $\vec{V}^{-1} = V_n^{-\ell_n} \cdot \cdots \cdot V_1^{-\ell_1}$ pour l'expression *inverse* de $\vec{V}$. En particulier, on a

$$\{\{U\}_{\vec{V}}\}_{\vec{V}^{-1}} = \{U\}_{\vec{V} \cdot \vec{V}^{-1}} \rightarrow_{\mathcal{R}}^* U$$

Nous étendons les notations applicables aux termes aux expressions $\vec{V}$ de la manière attendue : $\vec{V}\sigma = (V_1\sigma)^{\ell_1} \cdot \cdots \cdot (V_n\sigma)^{\ell_n}$, $\mathrm{var}(\vec{V}) = \bigcup_i \mathrm{var}(V_i)$ ...etc.

Enfin, un terme est *maximal* dans un ensemble de termes $\mathcal{S}$ ss'il n'est sous-terme strict d'aucun autre élément de $\mathcal{S}$.

6.4.2 Procédure de décision

Les tables 6.1 and 6.2 décrivent les règles de transformation utilisées pour simplifier pas à pas les problèmes d'équivalence ε. Nous voyons ici $\approx^?$ comme un symbole commutatif.

Dans chaque règle, le mot « frais » se rapporte à une variable ou à un nom de sorte appropriée (en l'occurrence toujours fixée par le contexte) et n'apparaissant pas dans l'équation de départ.

La procédure elle-même consiste à appliquer les règles décrites dans l'ordre suivant :

1. une séquence maximale de la règle **Undecipherable Encryption** ; puis
2. une séquence maximale de la règle **Split Names** ; puis

$$\frac{\varphi_1 \approx^? \varphi_2}{\varphi_1\{T \mapsto n\} \approx^? \varphi_2}$$

Undecipherable Encryption
si φ_1 est bien formée ; T est un sous-terme maximal de φ_1 de la forme $\mathsf{penc}(U, k', r)$ pour un certain nom k' (respectivement de la forme $\mathsf{pub}(k)$, $\mathsf{senc}(U, k, r)$, ou $\mathsf{enc}(U, k)$, pour un nom k dont toutes les occurrences dans φ_1 sont en position de clé) ; et n est un nom frais.

$$\frac{\varphi_1 \approx^? \varphi_2}{(\varphi_1\{n \mapsto \mathsf{pair}(n_1, n_2)\})\downarrow_{\mathcal{R}} \approx^? \varphi_2}$$

Split Names
si le nom n est de sorte $Pair[\tau_1, \tau_2]$; et n_1, n_2 sont des noms frais.

$$\frac{\varphi_1 \uplus \{x \mapsto \mathsf{pair}(U_1, V_1)\} \approx^? \varphi_2 \uplus \{x \mapsto \mathsf{pair}(U_2, V_2)\}}{\begin{array}{c}\varphi_1 \uplus \{y \mapsto U_1,\, z \mapsto V_1\} \\ \approx^? \varphi_2 \uplus \{y \mapsto U_2,\, z \mapsto V_2\}\end{array}}$$

Pair Analysis
si y, z sont des variables fraîches.

$$\frac{\varphi_1 \uplus \{x \mapsto U\} \approx^? \varphi_2 \uplus \{x \mapsto W\} \quad \text{notée } \varepsilon}{\varphi_1 \approx^? \varphi_2}$$

Redundancy Analysis-1
s'il existe M tel que
$$\begin{cases} \mathrm{var}(M) \subseteq \mathrm{dom}(\varphi_1) \\ \mathrm{names}(M) \cap \mathrm{names}(\varepsilon) = \emptyset \\ M\varphi_1 =_E U \\ W =_E M\varphi_2 \end{cases}$$

$$\frac{\varphi_1 \uplus \{x \mapsto U\} \approx^? \varphi_2 \uplus \{x \mapsto W\} \quad \text{notée } \varepsilon}{\bot}$$

Redundancy Analysis-2
s'il existe M tel que
$$\begin{cases} \mathrm{var}(M) \subseteq \mathrm{dom}(\varphi_1) \\ \mathrm{names}(M) \cap \mathrm{names}(\varepsilon) = \emptyset \\ M\varphi_1 =_E U \\ W \neq_E M\varphi_2 \end{cases}$$

Tab. 6.1 – Règles de transformation

$$\frac{\varphi_1 \uplus \{x \mapsto \{U\}_{\vec{V}}\} \approx^? \varphi_2 \uplus \{x \mapsto W\} \quad \text{notée } \varepsilon}{\varphi_1 \uplus \{y \mapsto S_1\} \approx^? \varphi_2 \uplus \{y \mapsto S_2\}}$$

Encryption Analysis-1
si $U = \mathsf{penc}(S_1, \mathsf{pub}(T_1), r_1)$ (ou respectivement $U = \mathsf{senc}(S_1, T_1, r_1)$) ;
r_1 n'apparaît nulle part ailleurs du même côté dans ε ;
il existe $\vec{M}$ et N tels que

$$\begin{cases} \mathrm{var}(\vec{M}, N) \subseteq \mathrm{dom}(\varphi_1) \\ \mathrm{names}(\vec{M}, N) \cap \mathrm{names}(\varepsilon) = \emptyset \\ \vec{M}\varphi_1 =_E \vec{V} \\ N\varphi_1 =_E T_1 \\ \{W\}_{\vec{M}^{-1}\varphi_2}\downarrow_{\mathcal{R}} \\ \quad = \mathsf{penc}(S_2, \mathsf{pub}(N\varphi_2)\downarrow_{\mathcal{R}}, r_2) \\ \text{(respectivement} \\ \quad = \mathsf{senc}(S_2, N\varphi_2\downarrow_{\mathcal{R}}, r_2)\ ) \end{cases}$$

r_2 n'apparaît nulle part ailleurs du même côté dans ε ; et y est une variable fraîche.

$$\frac{\varphi_1 \uplus \{x \mapsto \{U\}_{\vec{V}}\} \approx^? \varphi_2 \uplus \{x \mapsto W\} \quad \text{notée } \varepsilon}{\bot}$$

Encryption Analysis-2
si $U = \mathsf{penc}(S_1, \mathsf{pub}(T_1), r_1)$ (ou respectivement $U = \mathsf{senc}(S_1, T_1, r_1)$) ;
et il existe $\vec{M}$ et N tels que

$$\begin{cases} \mathrm{var}(\vec{M}, N) \subseteq \mathrm{dom}(\varphi_1) \\ \mathrm{names}(\vec{M}, N) \cap \mathrm{names}(\varepsilon) = \emptyset \\ \vec{M}\varphi_1 =_E \vec{V} \\ N\varphi_1 =_E T_1 \\ \mathsf{pdec\_success}(\{W\}_{\vec{M}^{-1}}, N)\varphi_2 \\ \quad \neq_E 1 \\ \text{(respectivement} \\ \mathsf{sdec\_success}(\{W\}_{\vec{M}^{-1}}, N)\varphi_2 \\ \quad \neq_E 1\ ) \end{cases}$$

$$\frac{\varphi_1 \uplus \{x \mapsto \{a\}_{\vec{C}[a_1,\ldots,a_n]}\} \approx^? \varphi_2}{\varphi_1\{a \mapsto \{a'\}_{\vec{C}[a_1,\ldots,a_n]^{-1}}\}\downarrow_{\mathcal{R}} \uplus \{x \mapsto a'\} \approx^? \varphi_2}$$

Standardize
si $a_1, \ldots, a_n \in \mathrm{im}(\varphi_1)$; $\vec{C} \neq \epsilon$; $a \notin \mathrm{im}(\varphi_1)$; et a' est un nom frais.

$$\frac{\varphi_1 \approx^? \varphi_2}{\varepsilon'}$$

Solve
si φ_1 et φ_2 sont directement transparents ; si $\varphi_1 \approx_E \varphi_2$ alors $\varepsilon' = \top$; sinon $\varepsilon' = \bot$.

TAB. 6.2 – Règles de transformation (suite)

3. une séquence maximale des règles d'analyse, c'est-à-dire de **Pair Analysis**, **Redundancy Analysis-{1,2}**, ou **Encryption Analysis-{1,2}** ; puis
4. une séquence maximale de la règle **Standardize** ; et enfin
5. une séquence maximale de la règle **Solve**.

Le but de la règle **Undecipherable Encryption** est de substituer un nom frais successivement pour chaque terme indéchiffrable T. Notons que toutes les occurrences de T sont remplacées à la fois. (Comme les environnements sont bien formés, et en particulier $\mathcal{R}$-réduits, un remplacement syntaxique suffit.)

La règle **Split Names** est une transformation technique utilisée pour supprimer les noms de sorte $Pair[\tau_1, \tau_2]$.

La règle **Pair Analysis** vise à casser les paires de termes conjointement dans les deux environnements testés.

Les deux règles **Redundancy Analysis-{1,2}** déterminent si une composante x d'un environnement est déductible des autres composantes. Si cette composante x se déduit de la même manière M des deux côtés alors x est supprimé partout (règle 1) ; sinon ceci fournit un contre-exemple à l'équivalence statique (règle 2).

Les deux règles **Encryption Analysis-{1,2}** sont destinées à détecter le succès du déchiffrement symétrique ou asymétrique dans une sous-composante, accessible sous plusieurs couches de chiffrement déterministe. Si, intuitivement, les deux environnements vérifient les mêmes relations, ainsi que différentes conditions techniques, alors les sous-composantes en question sont remplacées des deux côtés par les messages en clair correspondants. Dans le cas contraire, on peut conclure négativement.

Enfin, les deux règles **Standardize** et **Solve** concluent la procédure à l'aide de la notion d'environnements directement transparents.

Rappelons qu'un environnement φ est *directement transparent* (définition 5.8 du chapitre précédent) ssi $\mathrm{names}(\varphi) \subseteq \mathrm{im}(\varphi)$, c'est-à-dire si φ est de la forme

$$\varphi = \{x_1 \mapsto a_1, \ldots, x_m \mapsto a_m, y_1 \mapsto C_1[a_1, \ldots, a_m], \ldots, y_n \mapsto C_n[a_1, \ldots, a_m]\}$$

où les noms $a_1, \ldots, a_m$ sont mutuellement distincts.

D'après la proposition 5.14, deux environnements directement transparents ($i \in \{1, 2\}$)

$$\varphi_i = \{x_1^i \mapsto a_1^i, \ldots, x_{m_i}^i \mapsto a_{m_i}^i, y_1^i \mapsto C_1^i[a_1^i, \ldots, a_{m_i}^i], \ldots, y_{n_i}^i \mapsto C_{n_i}^i[a_1^i, \ldots, a_{m_i}^i]\}$$

sont statiquement équivalents ssi

$$\begin{array}{rrcl} & \{x_1^1, \ldots, x_{m_1}^1, y_1^1, \ldots, y_{n_1}^1\} & = & \{x_1^2, \ldots, x_{m_2}^2, y_1^2, \ldots, y_{n_2}^2\} \\ \text{pour tout } j \in \{1, \ldots, n_1\}, & y_j^1 \varphi_2 & =_E & C_j^1[x_1^1, \ldots, x_{m_1}^1]\varphi_2, \\ \text{et pour tout } k \in \{1, \ldots, n_2\}, & y_k^2 \varphi_1 & =_E & C_k^2[x_1^2, \ldots, x_{m_2}^2]\varphi_1. \end{array}$$

En utilisant ce critère, la règle **Solve** est donc bien effective.

Caractère effectif de la procédure

Il n'est pas strictement nécessaire pour notre preuve principale que les règles de transformation soient effectives, c'est-à-dire implémentable par un algorithme (terminant). Toutefois, nous justifions brièvement cette propriété de manière à obtenir une nouvelle procédure de décision pour les équivalences statiques considérées.

Le caractère effectif des règles **Undecipherable Encryption**, **Split Names**, **Pair Analysis**, et **Standardize** est clair, tandis que le cas de la règle **Solve** a été discuté plus haut. En ce qui concerne les règles **Encryption Analysis-{1,2}**, notre preuve de progression (plus bas) montre que pendant la procédure, une de ces règles s'applique toujours pourvu que

- $U = \mathsf{penc}(S_1, \mathsf{pub}(T_1), r_1)$ où r_1 n'apparaît nulle part ailleurs du même côté dans ε ; et
- il existe des termes P, N tels que $\mathrm{var}(P, N) \subseteq \mathrm{dom}(\varphi_1)$, $\mathrm{names}(P, N) \cap \mathrm{names}(\varepsilon) = \emptyset$, $P\varphi_1 =_E U$ et $N\varphi_1 = T_1$.

Trouver de tels termes U et N est du ressort des algorithmes classiques pour la déductibilité (rappelés dans la sous-section 6.4.3). Les règles **Redundancy Analysis-{1,2}** sont effectives pour des raisons similaires.

Exemples

Avant de débuter la preuve à proprement parler, nous illustrons la procédure sur quelques exemples.

Exemple 6.3. Considérons à nouveau l'environnement correspondant à une session du protocole Encrypted Key Exchange (EKE).

$$\varphi = \{x_1 \mapsto \{pk\}_{\mathsf{w}_{AB}},\ x_2 \mapsto \{\{k_1\}_{pk}^{r_1}\}_{\mathsf{w}_{AB}},\ x_3 \mapsto \{k_A\}_{k_1}^{r_2},\\ x_4 \mapsto \{\mathsf{pair}(k_A,\, k_B)\}_{k_1}^{r_3},\ x_5 \mapsto \{k_B\}_{k_1}^{r_4}\ \}$$

Soit $\varphi_i = \varphi\{\mathsf{w}_{AB} \mapsto \mathsf{c}_i\}$ pour $i \in \{0, 1\}$. Vérifions que $\varphi_0 \approx_E \varphi_1$ à l'aide de la procédure.

En effet, à partir de l'équation $\varepsilon_1 = (\varphi_0 \approx^? \varphi_1)$, nous pouvons appliquer la règle **Undecipherable Encryption** sur $T = \{k_1\}_{pk}^{r_1}$ de chaque côté, pour obtenir : (en omettant les restrictions)

$$\begin{aligned}\{x_1 \mapsto \{pk\}_{\mathsf{c}_0},\ x_2 \mapsto \{n_0\}_{\mathsf{c}_1},\ x_3 \mapsto \{k_A\}_{k_1}^{r_2},\\ x_4 \mapsto \{\mathsf{pair}(k_A,\, k_B)\}_{k_1}^{r_3},\ x_5 \mapsto \{k_B\}_{k_1}^{r_4}\ \}\\ \approx^? \{x_1 \mapsto \{pk\}_{\mathsf{c}_1},\ x_2 \mapsto \{n_0'\}_{\mathsf{c}_1},\ x_3 \mapsto \{k_A\}_{k_1}^{r_2},\\ x_4 \mapsto \{\mathsf{pair}(k_A,\, k_B)\}_{k_1}^{r_3},\ x_5 \mapsto \{k_B\}_{k_1}^{r_4}\ \}\end{aligned}$$

Ensuite, la même règle appliquée successivement sur les chiffrements par k_1 donne :

$$\begin{aligned}\{x_1 \mapsto \{pk\}_{\mathsf{c}_0},\ x_2 \mapsto \{n_0\}_{\mathsf{c}_1},\ x_3 \mapsto n_1,\ x_4 \mapsto n_2,\ x_5 \mapsto n_3\ \}\\ \approx^? \{x_1 \mapsto \{pk\}_{\mathsf{c}_1},\ x_2 \mapsto \{n_0'\}_{\mathsf{c}_1},\ x_3 \mapsto n_1',\ x_4 \mapsto n_2',\ x_5 \mapsto n_3'\ \}\end{aligned}$$

Comme aucune des règles **Analysis** ne s'applique, la règle **Standardize** sur pk, n_o et n_0' aboutit à :

$$\begin{aligned}\{x_1 \mapsto pk',\ x_2 \mapsto n_0'',\ x_3 \mapsto n_1,\ x_4 \mapsto n_2,\ x_5 \mapsto n_3\ \}\\ \approx^? \{x_1 \mapsto pk'',\ x_2 \mapsto n_0''',\ x_3 \mapsto n_1',\ x_4 \mapsto n_2',\ x_5 \mapsto n_3'\ \}\end{aligned}$$

On conclut alors positivement par la règle **Solve**. Dans cet exemple simple, les environnements finaux s'avèrent égaux modulo renommage. Nous verrons plus bas que ce n'est pas toujours le cas.

Exemple 6.4. Considérons à nouveau l'environnement correspondant à la variante modifiée du protocole EKE.

$$\varphi = \{x_1 \mapsto \{k_0\}_{\mathsf{w}_{AB}},\ x_2 \mapsto \{\{k_1\}_{k_0}^{r_1}\}_{\mathsf{w}_{AB}},\ x_3 \mapsto \{k_A\}_{k_1}^{r_2},\\ x_4 \mapsto \{\mathsf{pair}(k_A,\ k_B)\}_{k_1}^{r_3},\ x_5 \mapsto \{k_B\}_{k_1}^{r_4}\ \}$$

Soit $\varphi_i = \varphi\{\mathsf{w}_{AB} \mapsto \mathsf{c}_i\}$ $(i \in \{0,1\})$. Vérifions que notre procédure donne bien $\varphi_0 \not\approx_E \varphi_1$.

En effet, les environnements φ_i sont déjà simplifiés par rapport à **Undecipherable Encryption** et **Split Names**. En appliquant **Encryption Analysis-2** sur x_2, on obtient la réponse $\bot$. En effet, $x_2\varphi_0 = \{\mathsf{senc}(S_1, T_1, r_1)\}_{\mathsf{c}_0}$ où $T_1 =_E \mathsf{dec}(x_1, \mathsf{c}_0)\varphi_0$; mais

$$\mathsf{sdec\_success}(\mathsf{dec}(x_2, \mathsf{c}_0), \mathsf{dec}(x_1, \mathsf{c}_0))\varphi_1 \neq_E 1.$$

Exemple 6.5. Soit ε_3 l'équation suivante :

$$\{x_1 \mapsto \{\{n\}_k\}_k,\ x_2 \mapsto \{n\}_k,\ x_3 \mapsto \{k\}_{\mathsf{c}_0},\ x_4 \mapsto \{k\}_{\mathsf{c}_1}\ \}\\ \approx^? \{x_1 \mapsto \{n\}_k,\ x_2 \mapsto n,\ x_3 \mapsto \{k\}_{\mathsf{c}_0},\ x_4 \mapsto \{k\}_{\mathsf{c}_1}\ \}$$

La première règle applicable est **Standardize**. En appliquant cette règle à $x_2 \mapsto \{n\}_k$ pour le côté gauche puis à $x_3 \mapsto \{k\}_{\mathsf{c}_0}$ des deux côtés, on obtient :

$$\{x_1 \mapsto \{n'\}_{\mathsf{dec}(k',\mathsf{c}_0)},\ x_2 \mapsto n',\ x_3 \mapsto k',\ x_4 \mapsto \{\mathsf{dec}(k', \mathsf{c}_0)\}_{\mathsf{c}_1}\\ \approx^? \{x_1 \mapsto \{n\}_{\mathsf{dec}(k',\mathsf{c}_0)},\ x_2 \mapsto n,\ x_3 \mapsto k',\ x_4 \mapsto \{\mathsf{dec}(k', \mathsf{c}_0)\}_{\mathsf{c}_1}\ \}$$

ou encore, si l'on utilise la notation vectorielle :

$$\{x_1 \mapsto \{n'\}_{\{k'\}_{\mathsf{c}_0^{-1}}},\ x_2 \mapsto n',\ x_3 \mapsto k',\ x_4 \mapsto \{k'\}_{\mathsf{c}_0^{-1}\cdot\mathsf{c}_1}\ \}\\ \approx^? \{x_1 \mapsto \{n\}_{\{k'\}_{\mathsf{c}_0^{-1}}},\ x_2 \mapsto n,\ x_3 \mapsto k',\ x_4 \mapsto \{k'\}_{\mathsf{c}_0^{-1}\cdot\mathsf{c}_1}\ \}$$

Ces deux environnements sont égaux modulo renommage. Toutefois, si l'on choisit une stratégie différente à partir de ε_3, on peut également obtenir :

$$\{x_1 \mapsto n',\ x_2 \mapsto \{n'\}_{(\{k'\}_{\mathsf{c}_0^{-1}})^{-1}},\ x_3 \mapsto k',\ x_4 \mapsto \{k'\}_{\mathsf{c}_0^{-1}\cdot\mathsf{c}_1}\ \}\\ \approx^? \{x_1 \mapsto \{n\}_{\{k''\}_{\mathsf{c}_1^{-1}}},\ x_2 \mapsto n,\ x_3 \mapsto \{k''\}_{\mathsf{c}_1^{-1}\cdot\mathsf{c}_0},\ x_4 \mapsto k''\ \}$$

D'après le critère de la proposition 5.14, les deux environnements sont équivalents ssi les équations

$$x_1 \doteq \{x_2\}_{(\{x_4\}_{\mathsf{c}_1^{-1}})^{-1}} \qquad x_3 \doteq \{x_4\}_{\mathsf{c}_1^{-1}\cdot\mathsf{c}_0}$$

sont vérifiées également par l'environnement de gauche, tandis que les équations

$$x_2 \doteq \{x_1\}_{\{x_3\}_{\mathsf{c}_0^{-1}}} \qquad x_4 \doteq \{x_3\}_{\mathsf{c}_0^{-1}\cdot\mathsf{c}_1}$$

sont vérifiées également par l'environnement de droite.

Comme c'est le cas, la règle **Solve** s'applique pour donner $\top$. Notons que pour autant les deux derniers environnements ne sont pas égaux modulo renommage.

Nous illustrons maintenant le rôle des règles d'analyse.

Exemple 6.6. Soit ε_4 l'équation suivante :

$$\{x_1 \mapsto \{k_1\}_{\mathsf{c}_0},\ x_2 \mapsto \{\mathsf{pair}(k_2,k_3)\}^{r_0}_{k_1},\ x_3 \mapsto \{\{k_3\}_{\mathsf{c}_1}\}^{r_1}_{k_2}\ \} \\ \approx^? \{x_1 \mapsto \{k_1\}_{\mathsf{c}_0},\ x_2 \mapsto \{\mathsf{pair}(k_2,k_3)\}^{r_0}_{k_1},\ x_3 \mapsto \{\{k_4\}_{\mathsf{c}_1}\}^{r_1}_{k_2}\ \}$$

Soit $N = \mathsf{sdec}(x_2, \mathsf{dec}(x_1, \mathsf{c}_0))$. Comme le déchiffrement de x_3 par $\mathsf{fst}(N)$ réussit, et que r_1 n'apparaît pas ailleurs, la règle **Encryption Analysis-1** s'applique à deux reprises :

$$\{x_1 \mapsto \{k_1\}_{\mathsf{c}_0},\ x_2 \mapsto \{\mathsf{pair}(k_2,k_3)\}^{r_0}_{k_1},\ y_1 \mapsto \{k_3\}_{\mathsf{c}_1}\ \} \\ \approx^? \{x_1 \mapsto \{k_1\}_{\mathsf{c}_0},\ x_2 \mapsto \{\mathsf{pair}(k_2,k_3)\}^{r_0}_{k_1},\ y_1 \mapsto \{k_4\}_{\mathsf{c}_1}\ \}$$

Puis l'on applique successivement les règles **Encryption Analysis-1** et **Pair Analysis** sur x_2 :

$$\{x_1 \mapsto \{k_1\}_{\mathsf{c}_0},\ y_1 \mapsto \{k_3\}_{\mathsf{c}_1},\ y_2 \mapsto k_2,\ y_3 \mapsto k_3\} \\ \approx^? \{x_1 \mapsto \{k_1\}_{\mathsf{c}_0},\ x_3 \mapsto \{k_4\}_{\mathsf{c}_1}, y_2 \mapsto k_2,\ y_3 \mapsto k_3\ \}$$

suivies de **Standardize** sur x_1, et sur k_4 du côté droit :

$$\{x_1 \mapsto k'_1,\ y_1 \mapsto \{k_3\}_{\mathsf{c}_1},\ y_2 \mapsto k_2,\ y_3 \mapsto k_3\} \\ \approx^? \{x_1 \mapsto k'_1,\ y_1 \mapsto k'_4,\ y_2 \mapsto k_2,\ y_3 \mapsto k_3\ \}$$

Finalement, la règle **Solve** conclut $\bot$.

Enfin, notre dernier exemple illustre l'utilité des règles **Redundancy Analysis-{1,2}**.

Exemple 6.7. Soit ε_5 l'équation suivante :

$$\{x_1 \mapsto k_1,\ x_2 \mapsto k_2,\ x_3 \mapsto \{k_3\}^{r_0}_{k_1},\ x_4 \mapsto \{\{k_3\}^{r_0}_{k_1}\}_{k_2}\ \} \\ \approx^? \{x_1 \mapsto k_1,\ x_2 \mapsto k_2,\ x_3 \mapsto \{k_3\}^{r_0}_{k_1},\ x_4 \mapsto \{\{k_3\}^{r_1}_{k_1}\}_{k_2}\ \}$$

Remplacer $x_3 \mapsto \{k_3\}^{r_0}_{k_1}$ de chaque côté par $y_1 \mapsto k_3$ conduirait la procédure à répondre $\top$ de manière incorrecte. Ceci montre que l'on ne peut pas relâcher la condition de la règle **Encryption Analysis-1** concernant les occurrences des jetons probabilistes.

À l'inverse, notons que l'équation $x_3 \doteq \mathsf{pdec}(x_4, x_2)$ est vérifiée du côté gauche mais non du côté droit. Ainsi, la règle **Redundancy Analysis-2** s'applique pour conclure $\bot$.

Plan de la preuve

Les sous-sections suivantes constituent la preuve de la procédure proprement dite. Nous commençons (sous-section 6.4.3) par rappeler ou établir un certain nombre de résultats techniques sur la déductibilité pour la théorie équationnelle E considérée. Nous montrons ensuite successivement la terminaison et la progression (sous-section 6.4.4), la correction formelle (sous-section 6.4.5), et enfin la sûreté calculatoire de la procédure (sous-section 6.4.6).

6.4.3 Étude de la déductibilité

Nous énonçons ici plusieurs résultats techniques concernant la relation de déductibilité $\vdash_E$. Rappelons que par définition (section 5.1) $\varphi \vdash_E T$ ss'il existe un terme M tel que

- $\mathrm{var}(M) \subseteq \mathrm{dom}(\varphi)$,
- $\mathrm{names}(M) \cap \mathrm{names}(\varphi, T) = \emptyset$, et
- $M\varphi =_E T$.

Ainsi, nous gardons la convention du précédent chapitre consistant à restreindre implicitement tous les noms de φ *et* de T. La notation $\varphi \vdash_E T$ correspond à $\nu\tilde{n}.(\varphi \vdash T)$ où $\tilde{n} = \mathrm{names}(\varphi, T)$ chez M. Abadi et V. Cortier [AC06].

Caractérisation inductive

La proposition suivante, due à M. Abadi et V. Cortier [AC04], permet de caractériser la notion de déductibilité en terme de règles de déduction.

Proposition 6.8. *Soit E une théorie équationnelle engendrée par un système de réécriture sous-terme-convergent $\mathcal{R}$. On suppose que $\mathcal{R}$ ne contient aucun symbole privé (sans perte de généralité dans le cas présent où $\mathcal{F} = \mathcal{F}_{pub} \uplus \mathcal{N}$, par la proposition 2.9).*

Soit φ un environnement en forme $\mathcal{R}$-normale. On considère le plus petit ensemble $\mathrm{sat}(\varphi)$ *tel que*

(i) pour tout $x \in \mathrm{dom}(\varphi)$, $x\varphi \in \mathrm{sat}(\varphi)$;

(ii) pour tout $t \in \mathrm{st}(\varphi)$, si $t = f(t_1, \ldots, t_n)$ et $t_1, \ldots, t_n \in \mathrm{sat}(\varphi)$, alors $t \in \mathrm{sat}(\varphi)$;

(iii) pour toute règle $l \to r$ dans $\mathcal{R}$, s'il existe un contexte public C, des termes $t_1, \ldots, t_n \in \mathrm{sat}(\varphi)$, une substitution σ, et des noms $a_1, \ldots, a_m \notin \mathrm{names}(\varphi)$ tels que $l\sigma = C[t_1, \ldots, t_n, a_1, \ldots, a_m]$ et $r\sigma \in \mathrm{st}(\varphi)$, alors $r \in \mathrm{sat}(\varphi)$.

Un terme $\mathcal{R}$-réduit T est déductible de φ ss'il existe un contexte public C, des termes $t_1, \ldots, t_n \in \mathrm{sat}(\varphi)$ tels que $T = C[t_1, \ldots, t_n]$.

La preuve initiale de cette proposition est donnée dans un formalisme non typé [AC04] pour un système de réécriture $\mathcal{R}$ fini. Toutefois, cette preuve s'applique sans changement significatif au cas typé et pour un système de réécriture infini. Le résultat dans le cas infini et typé découle également de l'étude des règles de transformation introduites au chapitre 4.

Notons que lorsque $\mathcal{R}$ est sous-terme-convergent mais infini (comme c'est le cas ici), cette caractérisation n'implique pas immédiatement la décidabilité de la déductibilité, à l'inverse du cas fini [AC04]. En effet, pendant le calcul inductif de l'ensemble $\mathrm{sat}(\varphi)$, la condition (iii) implique un nombre potentiellement non borné de règles $(l \to r) \in \mathcal{R}$.

Nous considérons maintenant le cas particulier de la théorie équationnelle E

définie dans la section 6.3 :

$$\begin{aligned}
\mathsf{dec}_\tau(\mathsf{enc}_\tau(x,y),y) &\doteq x\\
\mathsf{enc}_\tau(\mathsf{dec}_\tau(x,y),y) &\doteq x\\
\mathsf{pdec}_\tau(\mathsf{penc}_\tau(x,\mathsf{pub}(y),z),y) &\doteq x\\
\mathsf{pdec_success}_\tau(\mathsf{penc}_\tau(x,\mathsf{pub}(y),z),y) &\doteq 1\\
\mathsf{sdec}_\tau(\mathsf{senc}_\tau(x,y,z),y) &\doteq x\\
\mathsf{sdec_success}_\tau(\mathsf{senc}_\tau(x,y,z),y) &\doteq 1\\
\mathsf{fst}_{\tau_1,\tau_2}(\mathsf{pair}_{\tau_1,\tau_2}(x,y)) &\doteq x\\
\mathsf{snd}_{\tau_1,\tau_2}(\mathsf{pair}_{\tau_1,\tau_2}(x,y)) &\doteq y\\
\mathsf{pair}_{\tau_1,\tau_2}(\mathsf{fst}_{\tau_1,\tau_2}(x),\mathsf{snd}_{\tau_1,\tau_2}(x)) &\doteq x
\end{aligned}$$

Appliquée à cette théorie et au système de réécriture $\mathcal{R}$ correspondant, la proposition 6.8 conduit à la caractérisation suivante.

Corollaire 6.9. *Soit φ un environnement en forme $\mathcal{R}$-normale. On considère le plus petit ensemble* $\mathrm{sat}(\varphi)$ *tel que*

(i) pour tout $x \in \mathrm{dom}(\varphi)$, $x\varphi \in \mathrm{sat}(\varphi)$;

(ii) pour tout $t \in \mathrm{st}(\varphi)$, si $t = f(t_1,\ldots,t_n)$ et $t_1,\ldots,t_n \in \mathrm{sat}(\varphi)$, alors $t \in \mathrm{sat}(\varphi)$;

(iii) les règles suivantes sont satisfaites : (les annotations de types étant omises)

$$\begin{aligned}
\mathsf{pair}(t_1,t_2) \in \mathrm{sat}(\varphi) &\Rightarrow t_1 \in \mathrm{sat}(\varphi)\\
\mathsf{pair}(t_1,t_2) \in \mathrm{sat}(\varphi) &\Rightarrow t_2 \in \mathrm{sat}(\varphi)\\
\mathsf{fst}(t) \in \mathrm{sat}(\varphi),\ \mathsf{snd}(t) \in \mathrm{sat}(\varphi)\ \textit{et}\ t \in \mathrm{st}(\varphi) &\Rightarrow t \in \mathrm{sat}(\varphi)\\
\mathsf{enc}(t_1,t_2) \in \mathrm{sat}(\varphi)\ \textit{et}\ t_2 \in \mathrm{sat}(\varphi) &\Rightarrow t_1 \in \mathrm{sat}(\varphi)\\
\mathsf{dec}(t_1,t_2) \in \mathrm{sat}(\varphi)\ \textit{et}\ t_2 \in \mathrm{sat}(\varphi) &\Rightarrow t_1 \in \mathrm{sat}(\varphi)\\
\mathsf{penc}(t_1,\mathsf{pub}(t_2),t_3) \in \mathrm{sat}(\varphi)\ \textit{et}\ t_2 \in \mathrm{sat}(\varphi) &\Rightarrow t_1 \in \mathrm{sat}(\varphi)\\
\mathsf{senc}(t_1,t_2,t_3) \in \mathrm{sat}(\varphi)\ \textit{et}\ t_2 \in \mathrm{sat}(\varphi) &\Rightarrow t_1 \in \mathrm{sat}(\varphi)
\end{aligned}$$

Un terme $\mathcal{R}$-réduit T est déductible de φ ss'il existe un contexte public C, des termes $t_1,\ldots,t_n \in \mathrm{sat}(\varphi)$ tels que $T = C[t_1,\ldots,t_n]$.

À la différence de la proposition 6.8, nous avons ici précisé l'instanciation du point (iii) par une analyse de cas sur les règles $\mathcal{R}$. Pour l'essentiel, on obtient ainsi le système classique de déduction à la Needham-Schroeder. Sans perte de généralité, nous avons omis les règles d'inférence laissant $\mathrm{sat}(\varphi)$ trivialement inchangé, par exemple :

$$t_1, t_2 \in \mathrm{sat}(\varphi) \text{ implique } t_1 \in \mathrm{sat}(\varphi) \text{ par } \mathsf{dec}(\mathsf{enc}(t_1,t_2),t_1) \to_{\mathcal{R}} t_1$$

tout comme celles redondantes avec le point (ii), par exemple :

$$\mathsf{penc}(t_1,\mathsf{pub}(t_2),t_3) \in \mathrm{sat}(\varphi) \text{ et } t_2 \in \mathrm{sat}(\varphi) \text{ implique } 1 \in \mathrm{sat}(\varphi).$$

Notons que $\mathrm{sat}(\varphi)$ est toujours inclus dans $\mathrm{st}(\varphi)$. Du corollaire 6.9, on déduit en particulier que la déductibilité dans E est décidable en temps polynomial en calculant $\mathrm{sat}(\varphi)$ par saturation. De plus, l'algorithme peut facilement retourner une recette dans le cas d'une réponse positive à $\varphi \vdash_E^? T$, autrement dit un terme M tel que $\mathrm{var}(M) \subseteq \varphi$, $\mathrm{names}(M) \cap \mathrm{names}(\varphi) = \emptyset$ et $M\varphi =_E T$. En effet, il suffit de

maintenir une table associant un tel terme à chaque élément de $\mathrm{sat}(\varphi)$. (Lorsqu'un élément est ajouté, on construit sans peine une recette pour cette élément à partir de la règle d'inférence utilisée, et des recettes existantes.)

Pour finir, nous spécialisons à nouveau cette caractérisation, dans le cas des environnements bien formés.

Corollaire 6.10. *Soit φ un environnement bien formé. Considérons le plus petit ensemble $\mathrm{sat}'(\varphi)$ tel que*

(i) pour tout $x \in \mathrm{dom}(\varphi)$, $x\varphi \in \mathrm{sat}'(\varphi)$;

(ii) les règles suivantes sont satisfaites : (les annotations de types étant omises)

$$\begin{aligned}
\mathsf{pair}(t_1,t_2) \in \mathrm{sat}'(\varphi) &\Rightarrow t_1 \in \mathrm{sat}'(\varphi)\\
\mathsf{pair}(t_1,t_2) \in \mathrm{sat}'(\varphi) &\Rightarrow t_2 \in \mathrm{sat}'(\varphi)\\
\mathsf{enc}(t_1,t_2) \in \mathrm{sat}'(\varphi) \text{ et } t_2 \in \mathrm{sat}'(\varphi) &\Rightarrow t_1 \in \mathrm{sat}'(\varphi)\\
\mathsf{penc}(t_1,\mathsf{pub}(t_2),t_3) \in \mathrm{sat}'(\varphi) \text{ et } t_2 \in \mathrm{sat}'(\varphi) &\Rightarrow t_1 \in \mathrm{sat}'(\varphi)\\
\mathsf{senc}(t_1,t_2,t_3) \in \mathrm{sat}'(\varphi) \text{ et } t_2 \in \mathrm{sat}'(\varphi) &\Rightarrow t_1 \in \mathrm{sat}'(\varphi)
\end{aligned}$$

Un terme $\mathcal{R}$-réduit T est déductible de φ ss'il existe un contexte public C, des termes $t_1,\dots,t_n \in \mathrm{sat}'(\varphi)$ tels que $T = C[t_1,\dots,t_n]$.

En effet, comme les clés de déchiffrement d'environnement bien formés sont toujours des noms, la règle d'inférence (ii) du corollaire 6.9 n'est pas nécessaire pour trouver de telles clés, et par conséquent peut être retardé. De plus, les environnements bien formés ne contiennent pas de symboles dec, fst et snd, si bien que les règles d'inférence correspondantes (troisième et sixième du point (iii) précédent) deviennent inutiles. Finalement, nous obtenons un système d'inférence très similaire à celui de M. Abadi et de P. Rogaway [AR02] ou encore de M. Abadi et B. Warinschi [AW05].

Quelques lemmes utiles

Fort de cette caractérisation de la déductibilité, nous pouvons énoncer maintenant plusieurs lemmes techniques utiles dans la suite de la preuve.

Étant donné un environnement φ et une expression $\vec{V} = V_1^{\ell_1} \cdot \dots \cdot V_n^{\ell_n}$, on note $\varphi \vdash_E \vec{V}$ lorsque pour tout $1 \le i \le n$, $\varphi \vdash_E V_i$.

Lemme 6.11. *Soit $\varphi_0 = \varphi \uplus \{x \mapsto \{U\}_{\vec{V}}\}$ un environnement bien formé.*

(1) On a $\varphi_0 \vdash_E \vec{V}$ ssi $\varphi \vdash_E \vec{V}$.

(2) Supposons $\varphi \vdash_E \vec{V}$. Alors pour tout terme T, $\varphi_0 \vdash_E T$ ssi $\varphi \uplus \{x \mapsto U\} \vdash_E T$.

Démonstration. (1) Supposons que $\varphi \vdash_E \vec{V}$: il existe $\vec{M}$ tel que $\mathrm{var}(\vec{M}) \subseteq \mathrm{dom}(\varphi)$, $\mathrm{names}(M) \cap \mathrm{names}(\varphi, T) = \emptyset$, et $\vec{M}\varphi =_E T$. Quitte à renommer les noms de $\mathrm{names}(\vec{M}) - \mathrm{names}(\varphi, T)$, on peut supposer sans perte de généralité que $\mathrm{names}(\vec{M}) \cap \mathrm{names}(\varphi_0) = \emptyset$. Ainsi, $\vec{M}$ montre que $\varphi_0 \vdash \vec{V}$.

Réciproquement, supposons que $\varphi \not\vdash_E \vec{V}$. Posons $\vec{V} = a_1^{\ell_1} \cdots a_m^{\ell_m}$. Comme φ est bien formée, on a pour tout i, $\ell_i = 1$; soit $\vec{V} = a_1 \cdots a_m$. Par hypothèse,

il existe un indice maximal $i \geq 1$ tel que $\varphi \not\vdash_E a_i$. D'après le corollaire 6.10, on a

$$\mathrm{sat}'(\varphi_0) = \mathrm{sat}'(\varphi) \cup \left\{ \{U\}_{\vec{V}'} \mid \vec{V}' = a_1 \cdots a_j,\ j \geq i \right\}.$$

En particulier, comme $a_i \notin \mathrm{sat}'(\varphi)$, on a $\varphi_0 \not\vdash_E \vec{V}$.

(2) Soit $\vec{N}\varphi =_E \vec{V}$ avec $\mathrm{var}(N) \subseteq \mathrm{dom}(\varphi)$ et $\mathrm{names}(N) \cap \mathrm{names}(\varphi_0, T) = \emptyset$.
Supposons que l'on ait $M\varphi_0 =_E T$ avec $\mathrm{var}(M) \subseteq \mathrm{dom}(\varphi_0)$ et $\mathrm{names}(M) \cap \mathrm{names}(\varphi_0, T) = \emptyset$. Alors $M\{x \mapsto \{x\}_{\vec{N}}\}(\varphi \uplus \{x \mapsto U\}) =_E T$, d'où $\varphi \uplus \{x \mapsto U\} \vdash_E T$.
Réciproquement, supposons que $M(\varphi \uplus \{x \mapsto U\}) =_E T$ avec $\mathrm{var}(M) \subseteq \mathrm{dom}(\varphi_0)$ et $\mathrm{names}(M) \cap \mathrm{names}(\varphi_0, T) = \emptyset$. On a alors l'égalité $M\{x \mapsto \{x\}_{\vec{N}^{-1}}\}\varphi_0 =_E T$, d'où $\varphi_0 \vdash_E T$. □

Lemme 6.12. *Soit $\varphi_0 = \varphi \uplus \{x \mapsto \mathsf{penc}(U, \mathsf{pub}(V), r)\}$ un environnement bien formé (respectivement $\varphi_0 = \varphi \uplus \{x \mapsto \mathsf{senc}(U, V, r)\}$).*

(1) On a $\varphi_0 \vdash_E V$ ssi $\varphi \vdash_E V$.

(2) Supposons que $\varphi \vdash_E V$ et que r n'apparaît pas dans φ. Pour tout terme T qui ne contient pas r, $\varphi_0 \vdash_E T$ ssi $\varphi \uplus \{y \mapsto U\} \vdash_E T$.

Démonstration. Nous considérons que le cas du chiffrement asymétrique uniquement, dans la mesure où le cas symétrique se traite de manière identique.

(1) Similaire au lemme 6.11.

(2) Soit T tel que $r \notin \mathrm{names}(T)$. Posons $N\varphi =_E V$ avec $\mathrm{var}(N) \subseteq \mathrm{dom}(\varphi)$ et $\mathrm{names}(N) \cap \mathrm{names}(\varphi_0, T) = \emptyset$.
Supposons que l'on ait $M\varphi_0 =_E T$ avec $\mathrm{var}(M) \subseteq \mathrm{dom}(\varphi_0)$ et $\mathrm{names}(M) \cap \mathrm{names}(\varphi_0, T) = \emptyset$. Soit r' un nom frais de sorte *Coins*. Comme r m'apparaît pas dans φ ni dans T, on a $M\{x \mapsto \{y\}_N^{r'}\}(\varphi \uplus \{y \mapsto U\}) =_E T\{r \mapsto r'\} = T$, d'où $\varphi \uplus \{y \mapsto U\} \vdash_E T$.
Réciproquement, supposons que l'on ait $M(\varphi \uplus \{y \mapsto U\}) =_E T$ avec $\mathrm{var}(M) \subseteq \mathrm{dom}(\varphi_0)$ et $\mathrm{names}(M) \cap \mathrm{names}(\varphi_0, T) = \emptyset$. On a alors $M\{x \mapsto \mathsf{pdec}(x, N)\}\varphi =_E T$, d'où $\varphi_0 \vdash_E T$. □

Définition 6.5. Une position p est *sous un chiffrement opaque* dans φ ss'il existe une variable $x \in \mathrm{dom}(\varphi)$ et un préfixe $p' \cdot 1$ de p tels que $(x\varphi)|_{p'}$ est de la forme
- $\mathsf{enc}(U, V)$ avec $\varphi \not\vdash_E V$,
- $\mathsf{senc}(U, V, W)$ avec $\varphi \not\vdash_E V$, ou
- $\mathsf{penc}(U, V', W)$ avec $\forall V,\ V' =_E \mathsf{pub}(V) \Rightarrow \varphi \not\vdash_E V$.

L'expression $p' \cdot 1$ signifie que p se trouve sous le premier argument U du chiffrement.

Lemme 6.13. *Soit φ un environnement bien formé, et $r \in \mathrm{names}(\varphi)$ de sorte Coins. Alors, r n'est pas déductible de φ.*

Démonstration. Comme φ est bien formé, r apparaît uniquement comme troisième argument d'un symbole de chiffrement. Par inspection des règles du corollaire 6.10, on montre que $r \notin \mathrm{sat}'(\varphi)$. Ainsi r n'est pas déductible. □

Lemme 6.14. *Soit φ un environnement bien formé, et T un sous-terme de φ de sorte $\tau \neq$ Coins. Alors, T est déductible de φ ssi ou bien (i) il existe un symbole*

f tel que $T = f(T_1, \ldots, T_n)$ et tous les T_i sont déductibles, ou bien (ii) T apparaît à une position p de φ qui n'est pas une position de clé ni ne se trouve sous un chiffrement opaque.

Démonstration. Grâce au corollaire 6.10, il suffit de vérifier que (ii) est vraie ssi $T \in \mathrm{sat}'(\varphi)$.

Nous montrons l'implication $T \in \mathrm{sat}'(\varphi) \Rightarrow$ (ii) par induction sur l'arbre de preuve pour $T \in \mathrm{sat}'(\varphi)$.

- Si le point (i) du corollaire 6.10 s'applique (règle axiome), alors T apparaît à la racine, ce qui satisfait (ii).
- Si la règle $\mathsf{pair}(t_1, t_2) \in \mathrm{sat}'(\varphi) \Rightarrow t_i \in \mathrm{sat}'(\varphi)$ s'applique avec $t_i = T$, alors l'hypothèse d'induction s'applique à $\mathsf{pair}(t_1, t_2)$ ce qui implique (ii) sur T.
- Si la règle $\mathsf{enc}(t_1, t_2) \in \mathrm{sat}'(\varphi)$ et $t_2 \in \mathrm{sat}'(\varphi) \Rightarrow t_1 \in \mathrm{sat}'(\varphi)$ s'applique avec $t_1 = T$, alors l'hypothèse d'induction s'applique à $\mathsf{enc}(t_1, t_2)$ entraînant (ii) sur T puisque t_2 est déductible.
- (Les cas des règles restantes sont similaires.)

Quant à l'implication réciproque, nous la prouvons par induction sur la position p de T dans φ, donnée par (ii). En effet, si p est à une racine de φ, la règle axiome (i) s'applique. Supposons que $p = p' \cdot i$. Comme φ est bien formé, que T est de sorte $\tau \neq \mathit{Coins}$ et que (ii) est vérifiée, on obtient

- le symbole en position p' de φ est ou bien pair, enc, penc ou senc ;
- dans les trois derniers cas, T est le premier argument du chiffrement et le second argument correspond à une clé de déchiffrement déductible ;
- p' n'est ni en position de clé, ni sous un chiffrement opaque dans φ.

En appliquant l'hypothèse d'induction à T' et la règle d'inférence appropriée du corollaire 6.10, on obtient que T appartient à $\mathrm{sat}'(\varphi)$. □

6.4.4 Terminaison et progression

Tout d'abord, nous vérifions que

(i) toutes les règles de transformation excepté **Standardize** préservent la bonne formation des environnements ;

(ii) les règles d'une même séquence, excepté **Standardize** et **Solve**, terminent sur des environnements bien formés, tandis que **Standardize** et **Solve** terminent dans le cas général.

Ces propriétés sont claires pour la plupart des règles.

- Concernant **Split Names**, chaque application de la règle fait décroître strictement la somme
$$\sum_{n \in \mathrm{names}(\varepsilon)} |\mathrm{sort}(n)|$$
où la notation $|\tau|$ désigne la taille (sans partage) du type τ.
- Dans le cas de **Encryption Analysis**, supposons les deux côtés de l'équation ε bien formés. Alors, étant donné que W est $\mathcal{R}$-réduit, la relation
$$\{W\}_{\vec{M}^{-1}\varphi_2}\downarrow_{\mathcal{R}} = \mathsf{penc}(S_2, \mathsf{pub}(N\varphi_2)\downarrow_{\mathcal{R}}, r_2)$$
est équivalente à
$$W = \{\mathsf{penc}(S_2, \mathsf{pub}(N\varphi_2)\downarrow_{\mathcal{R}}, r_2)\}_{\vec{M}\varphi_2\downarrow_{\mathcal{R}}}$$

(et de même pour le chiffrement symétrique). Ainsi, S_2 est un sous-terme du membre droit.
– Concernant la terminaison de **Standardize**, notons que chaque application de cette règle fait décroître la taille de $\text{names}(\varphi) - \text{im}(\varphi)$, où φ est le côté de l'équation considéré.

Pour montrer la progression, nous prouvons successivement les faits suivants :

(1) pour toute équation $\varphi_1 \approx^? \varphi_2$ atteinte après la séquence 1 (règle **Undecipherable Encryption**), tous les sous-termes T de φ_i de sorte $\tau \neq \mathit{Coins}$ sont déductibles de φ_i, et φ_i ne contient pas de sous-terme de la forme $\mathsf{penc}(U, k', r)$;

(2) pour toute équation $\varphi_1 \approx^? \varphi_2$ atteinte après la séquence 2 (règle **Split Names**), φ_1 et φ_2 ne contiennent aucun nom de sorte $\mathit{Pair}[\tau_1, \tau_2]$;

(3) pour toute équation $\varphi_1 \approx^? \varphi_2$ atteinte après la séquence 3 (règles **Pair**, **Redundancy** et **Encryption Analysis**), φ_1 et φ_2 ne contiennent aucun symbole pair, penc, ou senc ;

(4) pour toute équation $\varphi_1 \approx^? \varphi_2$ atteinte après la séquence 4 (règle **Standardize**), φ_1 et φ_2 sont directement transparents.

Étant donné la définition de la règle **Solve**, le dernier point implique que lorsque les séquences à 1 à 5 sont terminées, alors l'équation résultante est $\bot$ ou $\top$.

Nous établissons maintenant les quatre points. Dans la suite de la preuve, $\varepsilon = \varphi_1^0 \approx^? \varphi_2^0$ désigne l'équation de départ de la règle considérée tandis que l'équation d'arrivée est notée $\varepsilon' = \varphi_1^1 \approx^? \varphi_2^1$.

(1) Tout d'abord, nous vérifions que les règles des séquences 2 à 5 ne créent pas de sous-termes non déductibles. (Clairement, ils ne créent pas de termes de la forme $\mathsf{penc}(U, k', r)$.) Concernant **Redundancy Analysis-1**, comme les composantes supprimées U et W restent déductibles de leurs environnements respectifs, c'est le cas également des termes restants. Concernant **Encryption Analysis-1**, soit T un sous-terme de l'environnement résultant φ_i^1. Grâce à la condition sur les jetons r_i de φ_i^0, r_i n'apparaît pas dans T. Comme $\vec{V}$ et T sont déductibles de φ, les lemmes 6.11 et 6.12 s'appliquent : on obtient que T est déductible de $\varphi_i^1 = \varphi_i \uplus \{y \mapsto S_i\}$. Le cas des autres règles est clair.

Deuxièmement, nous montrons la propriété de progression pour la règle **Undecipherable Encryption**. Soit $\varphi_1^0 \approx^? \varphi_2^0$ une équation atteinte juste après la séquence 1. Par contradiction, supposons qu'il existe un sous-terme $\mathsf{penc}(T, k', r)$ ou un sous-terme non déductible T de sorte $\tau \neq \mathit{Coins}$ dans φ_1. Dans le second cas, par le lemme 6.14, T apparaît ou bien en position de clé, ou bien sous un chiffrement opaque.

Ceci implique que φ_1^0 contient ou bien un sous-terme $\mathsf{penc}(T, k', r)$, ou bien un nom k de sorte $\tau \neq \mathit{EKey}$, en position de clé et non déductible de φ_1. Dans le premier cas, la règle **Undecipherable Encryption** s'applique. Dans le second cas, du fait de la condition d'acyclicité, il existe un nom k_0, non déductible, de sorte $\tau \neq \mathit{EKey}$, en position de clé, et maximal pour l'ordre des clés, c'est-à-dire, tel que k_0 n'apparaît (excepté les position de clé) chiffré sous aucune clé. Comme k_0 n'est pas déductible, par le lemme 6.14, il apparaît uniquement en position de clé. La règle **Undecipherable Encryption** s'applique alors à un sous-terme maximal T de la forme $\mathsf{pub}(k_0)$, $\mathsf{senc}(U, k_0, r)$ ou $\mathsf{enc}(U, k_0)$.

(2) D'après leurs définitions, les règles des séquences 3 à 5 ne créent pas de noms de sorte $Pair[\tau_1, \tau_2]$. De plus, la règle **Split Names** termine lorsque de tels noms n'apparaissent plus.

(3) Les séquences 3 à 5 ne créent pas de symboles pair, penc et senc. Soit $\varphi_1^0 \approx^? \varphi_2^0$ une équation atteinte juste après la séquence 3. Par contradiction, soit U un sous-terme maximal de φ_1^0 (par exemple) tel que le symbole de tête de U est pair, penc, ou senc. Étant donné le système de sortes, et comme, par bonne formation et par (2), les autres symboles disponibles sont enc, pub, les constantes et les noms, il existe x dans $\mathrm{dom}(\varphi_1^0)$ tel que $x\varphi_1^0$ est de la forme $\{U\}_{\vec{V}}$ (éventuellement $\vec{V} = \epsilon$). Soit $\varphi_1^0 = \varphi_1 \uplus \{x \mapsto \{U\}_{\vec{V}}\}$.

Si U est une paire, alors comme $T_{\mathsf{enc}} \cap \{Pair[\tau_1, \tau_2]\}_{\tau_1,\tau_2} = \emptyset$, on a $\vec{V} = \epsilon$. Pour la même raison, et en utilisant (2), le symbole de tête de $x\varphi_2^0$ est pair ; par conséquent, la règle **Pair Analysis** s'applique.

Dans le cas contraire, par (1) et (2) on peut supposer par exemple que U est de la forme $\mathsf{penc}(S_1, \mathsf{pub}(T_1), r_1)$. (Le cas du chiffrement symétrique est similaire.) Par (1), T_1 et $\vec{V}$ sont déductibles de φ_1^0. Par les lemmes 6.11 et 6.12, ceci implique que T_1 et $\vec{V}$ sont déductibles de φ_1 lui-même. Soit $\vec{M}$ et M tels que $\mathrm{var}(\vec{M}, N) \subseteq \mathrm{dom}(\varphi)$, $\mathrm{names}(\vec{M}, N) \cap \mathrm{names}(\varepsilon) = \emptyset$ (par stabilité de E par renommage on peut exclure un nombre fini de noms dans $\vec{M}$ et N), $\vec{M}\varphi_1 =_E \vec{V}$ et $N\varphi =_E T_1$.

Nous distinguons trois cas.

- Si U est déductible de φ_1, c'est-à-dire, s'il existe P tel que $\mathrm{var}(P) \subseteq \mathrm{dom}(\varphi)$, $\mathrm{names}(P) \cap \mathrm{names}(\varepsilon) = \emptyset$ et $P\varphi_1 =_E U$, l'une des deux règles **Redundancy Analysis-1,2** s'applique, selon que l'équation $x =_E \{P\}_{\vec{M}}$ est vérifiée dans φ_2^0 ou non ; contradiction.
- Si la différence $\mathsf{pdec\_success}(\{x\}_{\vec{M}^{-1}}, N)\varphi_2^0 \neq_E 1$ est vraie, alors la règle **Encryption Analysis-2** s'applique.
- Dans le cas contraire, le terme U n'est pas déductible de l'environnement φ_1 et l'on a $\mathsf{pdec\_success}(\{x\}_{\vec{M}^{-1}}, N)\varphi_2 =_E 1$. Pour commencer, nous montrons que r_1 n'apparaît pas ailleurs dans φ_1^0.
  En effet, par contradiction, supposons que r_1 apparaisse ailleurs dans φ_1^0. Par bonne formation, r_1 apparaît uniquement en position $r_1 = U|_3$ dans φ_1^0, et U n'est pas sous-terme de $\vec{V}$. Par conséquent, U est un sous-terme de φ_1. Comme U n'est pas déductible de φ_1, par le lemme 6.14 et la propriété (1), U apparaît dans φ_1 sous un chiffrement par une clé k_1 de sorte différente de *EKey*, non déductible de φ_1. De plus, par (1), k_1 est déductible de φ_1^0. Par le lemme 6.14, on en déduit que k_1 apparaît dans φ_1^0 non en position de clé. Comme k_1 chiffre U, par bonne formation, cette occurrence doit être dans φ_1 (elle ne peut pas être dans U ni dans $\vec{V}$ sinon ceci formerait un cycle de chiffrement). En appliquant le même raisonnement à k_1 et ainsi de suite, on obtient l'existence d'une séquence infinie de noms $k_1, k_2, \ldots, k_i, \ldots$ de sortes différentes de *EKey*, tels que pour tout $i \geq 1$, k_i chiffre k_{i-1}, apparaît dans φ_1 non en position de clé, et n'est pas déductible dans φ_1. Comme $\mathrm{names}(\varphi_1)$ est fini, ceci contredit la condition d'acyclicité.
  Ainsi, r_1 n'apparaît nulle part ailleurs dans φ_1^0. Soit $U_2 = \{W\}_{\vec{M}^{-1}\varphi_2}\downarrow_{\mathcal{R}}$. De l'équation $\mathsf{pdec\_success}(\{x\}_{\vec{M}^{-1}}, N)\varphi_2 =_E 1$ et de la bonne formation de φ_1^0, on déduit que U_2 est de la forme $\mathsf{penc}(S_2, \mathsf{pub}(T_2), r_2)$. De manière similaire

à U_1, on montre que U_2 est déductible de φ_2^0 (ainsi la règle **Redundancy Analysis** s'applique), ou bien r_2 apparaît une fois dans φ_2^0. Dans le premier cas, la règle **Undecipherable Encryption** s'applique ; contradiction.

(4) Soit $\varphi_1^0 \approx^? \varphi_2^0$ une équation atteinte juste après la séquence 4. Par contradiction, supposons que, par exemple, φ_1^0 n'est pas directement transparent. Autrement dit, l'ensemble $\text{names}(\varphi_1^0) - \text{im}(\varphi_1^0)$ est non vide.

Par (3), les seuls symboles de φ_1^0 sont les constantes $(0, 1, \mathsf{w}, \ldots)$, les noms et les symboles enc, dec et pub. Étant donné le système de sorte, ceci implique que pour tout x dans $\text{dom}(\varphi_1^0)$, $x\varphi_1^0$ de la forme

$$x\varphi_1^0 = \begin{cases} & \{c\}_{\vec{C}[a_1,\ldots,a_n]} \\ \text{ou} & \{a\}_{\vec{C}[a_1,\ldots,a_n]} \\ \text{ou} & \{\mathsf{pub}(k)\}_{\vec{C}[a_1,\ldots,a_n]} \end{cases}$$

où c désigne une constante arbitraire, a est un nom, $\vec{C}$ un contexte public vectoriel (éventuellement vide) fait de constantes et de symboles enc et dec uniquement, et où les a_i sont des noms de sorte *Data*.

Étant donné la forme de l'environnement, la procédure de saturation décrite dans le corollaire 6.9 se réduit aux règles suivantes :

(i) pour tout $x \in \text{dom}(\varphi_1^0)$, $x\varphi \in \text{sat}(\varphi_1^0)$; autrement dit, $\text{im}(\varphi_1^0) \subseteq \text{sat}(\varphi_1^0)$;

(ii) pour tout $t \in \text{st}(\varphi_1^0)$, si $t = f(t_1, \ldots, t_n)$ et $t_1, \ldots, t_n \in \text{sat}(\varphi_1^0)$, alors $t \in \text{sat}(\varphi_1^0)$;

(iii) si $x\varphi_1^0 = \{a\}_{\vec{C}[a_1,\ldots,a_n]}$, et $a_1, \ldots, a_n \in \text{sat}(\varphi_1^0)$ alors pour tout $\vec{C}' \in \text{st}(\vec{C})$ (éventuellement vide), on a $\{a\}_{\vec{C}'[a_1,\ldots,a_n]} \in \text{sat}(\varphi_1^0)$.

Notons qu'en ce qui concerne la déductibilité des noms,
- la règle (i) peut être restreinte aux noms dans $\text{im}(\varphi_1^0)$;
- la règle (ii) est inutile ;
- et la règle (iii) peut être restreinte à $\vec{C}' = \epsilon$, c'est-à-dire à la conclusion $a \in \text{sat}(\varphi_1^0)$.

Comme, par (1), tous les noms de φ_1^0 sont déductibles, $\text{names}(\varphi_1^0)$ est le plus petit ensemble saturé par les règles (i) et (iii) restreintes aux noms. En particulier, comme $\text{names}(\varphi_1^0) - \text{im}(\varphi_1^0) \neq \emptyset$, il existe un nom $a \in \text{names}(\varphi_1^0) - \text{im}(\varphi_1^0)$ tel que la dérivation correspondante par (i) et (iii) est minimale. Comme $a \notin \text{im}(\varphi_1^0)$, cette dérivation se termine par (iii). Ainsi, il existe une variable x tels que $x\varphi_1^0 = \{a\}_{\vec{C}[a_1,\ldots,a_n]}$ et $a_1, \ldots, a_n \in \text{im}(\varphi_1^0)$, et par conséquent, la règle **Standardize** s'applique.

6.4.5 Correction formelle

Comme ci-dessus, l'expression $\varepsilon = \varphi_1^0 \approx^? \varphi_2^0$ désigne l'équation de départ de chaque règle considérée tandis que l'équation résultante est notée $\varepsilon' = \varphi_1^1 \approx^? \varphi_2^1$.

Nous vérifions maintenant pour chaque règle que ε est vraie formellement ssi ε' l'est. Dans la définition de l'équivalence statique :

$\varphi_1 \approx_E \varphi_2$ ssi pour tous M, N tels que $\text{var}(M, N) \subseteq \text{dom}(\varphi_1) = \text{dom}(\varphi_2)$ et $\text{names}(M, N) \cap \text{names}(\varphi_1, \varphi_2)$, on a

$$M\varphi_1 =_E N\varphi_1 \Leftrightarrow M\varphi_2 =_E N\varphi_2,$$

notons que l'on peut interdire sans perte de généralité un nombre fini, arbitraire de noms dans M et N (en plus de $\mathrm{names}(\varphi_1, \varphi_2)$), car E est stable par renommage des noms.

- **Undecipherable Encryption**. Montrons que $\varphi_1 \approx_E \varphi_1^1 = \varphi_1\{T \mapsto n\}$. Pour cela, sachant que $\mathcal{R}$ est sous-terme-convergent, et que T et φ_1 sont $\mathcal{R}$-réduits, nous vérifions les hypothèses restantes de la proposition 6.4 selon les différents cas.
  - Si $T = \mathsf{penc}(U, k', r)$, sachant que r n'est pas déductible (lemme 6.13), et que pour tout V, $k' \neq_E \mathsf{pub}(V)$ (par inspection des règles du système convergent $\mathcal{R}$), la proposition 6.4 s'applique bien à T et φ_1, pour $i = 3$ et $\mathcal{R}_0 = \mathcal{R} - \mathcal{R}_3$.
  - Si $T = \mathsf{pub}(k)$ et si les seules occurrences de k dans φ^0 sont en position de clé, alors par le lemme 6.14, on a $\varphi_1 \not\vdash_E k$: la proposition 6.4 s'applique donc avec $i = 4$ et $\mathcal{R}_0 = \mathcal{R} - \mathcal{R}_4$.
  - Les cas $T = \mathsf{senc}(U, k, r)$ et $T = \mathsf{enc}(U, k)$ se traitent de la même façon, respectivement pour $i = 2$ et $i = 1$.
- **Split Names**. Montrons que $\varphi_1 \approx_E \varphi_1^1 = \varphi_1\{n \mapsto \mathsf{pair}(n_1, n_2)\downarrow_{\mathcal{R}}\}$.
  Soit M, N deux termes tels que $\mathrm{names}(M, N) \cap \mathrm{names}(\varphi_1, \varphi_1^1) = \emptyset$. Supposons que $M\varphi_1 =_E N\varphi_1$. Alors

$$\begin{aligned} M\varphi_1^1 &=_E (M\varphi_1)\{n \mapsto \mathsf{pair}(n_1, n_2)\} \\ &=_E (N\varphi_1)\{n \mapsto \mathsf{pair}(n_1, n_2)\} =_E N\varphi_1^1 \end{aligned}$$

  Réciproquement, supposons que $M\varphi_1^1 =_E N\varphi_1^1$. Alors

$$\begin{aligned} M\varphi_1 &=_E (M\varphi_1)\{n_1 \mapsto \mathsf{fst}(n),\, n_2 \mapsto \mathsf{snd}(n)\} \\ &=_E (N\varphi_1^1)\{n_1 \mapsto \mathsf{fst}(n),\, n_2 \mapsto \mathsf{snd}(n)\} \\ &=_E N\varphi_1^1 \end{aligned}$$

- **Pair Analysis**. Supposons que $\varphi_1^0 \approx_E \varphi_2^0$. Soit P, Q deux termes tels que $\mathrm{var}(P, Q) \subseteq \mathrm{dom}(\varphi_1^1) = \mathrm{dom}(\varphi_2^1)$, et $\mathrm{names}(P, Q) \cap \mathrm{names}(\varphi_1^1, \varphi_2^1) = \emptyset$. Étant donné que

$$P\varphi_i^1 =_E P\{y \mapsto \mathsf{fst}(x),\, z \mapsto \mathsf{snd}(x)\}\varphi_1^0$$
$$\mathrm{names}(P\{y \mapsto \mathsf{fst}(x),\, z \mapsto \mathsf{snd}(x)\}) \cap \mathrm{names}(\varphi_1^0, \varphi_2^0) = \emptyset$$

  (et de même manière pour Q), on a

$$\begin{aligned} & P\varphi_1^1 =_E Q\varphi_1^1 \\ \Leftrightarrow \quad & P\{y \mapsto \mathsf{fst}(x),\, z \mapsto \mathsf{snd}(x)\}\varphi_1^0 =_E Q\{y \mapsto \mathsf{fst}(x),\, z \mapsto \mathsf{snd}(x)\}\varphi_1^0 \\ \Leftrightarrow \quad & P\{y \mapsto \mathsf{fst}(x),\, z \mapsto \mathsf{snd}(x)\}\varphi_2^0 =_E Q\{y \mapsto \mathsf{fst}(x),\, z \mapsto \mathsf{snd}(x)\}\varphi_2^0 \\ \Leftrightarrow \quad & P\varphi_2^1 =_E Q\varphi_2^1 \end{aligned}$$

  Réciproquement, supposons que $\varphi_1^1 \approx_E \varphi_2^1$. Soit P, Q deux termes tels que $\mathrm{var}(P, Q) \subseteq \mathrm{dom}(\varphi_1^0) = \mathrm{dom}(\varphi_2^0)$, et $\mathrm{names}(P, Q) \cap \mathrm{names}(\varphi_1^0, \varphi_2^0) = \emptyset$. Étant donné que

$$P\varphi_i^0 =_E P\{x \mapsto \mathsf{pair}(y, z)\}\,\varphi_i^1$$
$$\mathrm{names}(P\{x \mapsto \mathsf{pair}(y, z)\}) \cap \mathrm{names}(\varphi_1^1, \varphi_2^1) = \emptyset$$

(et de même pour Q), on a

$$\begin{array}{rc} & P\varphi_1^0 =_E Q\varphi_1^0 \\ \Leftrightarrow & P\{x \mapsto \mathsf{pair}(y,z)\}\varphi_1^1 =_E Q\{x \mapsto \mathsf{pair}(y,z)\}\varphi_1^1 \\ \Leftrightarrow & P\{x \mapsto \mathsf{pair}(y,z)\}\varphi_2^1 =_E Q\{x \mapsto \mathsf{pair}(y,z)\}\varphi_2^1 \\ \Leftrightarrow & P\varphi_2^0 =_E Q\varphi_2^0 \end{array}$$

– **Redundancy Analysis-2**. Sachant que le test $x =^? M$ est vérifié dans φ_1^0 mais non dans φ_2^0, on a bien $\varphi_1^0 \not\approx_E \varphi_2^0$.

– **Redundancy Analysis-1**. Supposons que $\varphi_1^0 \approx_E \varphi_2^0$. Soit Q_1, Q_2 deux termes tels que $\mathrm{var}(Q_1,Q_2) \subseteq \mathrm{dom}(\varphi_1) = \mathrm{dom}(\varphi_2)$, et $\mathrm{names}(Q_1,Q_2) \cap \mathrm{names}(\varphi_1^0,\varphi_2^0) = \emptyset$. On a

$$\begin{array}{rc} & Q_1\varphi_1 =_E Q_2\varphi_1 \\ \Leftrightarrow & Q_1\varphi_1^0 =_E Q_2\varphi_1^0 \\ \Leftrightarrow & Q_1\varphi_2^0 =_E Q_2\varphi_2^0 \\ \Leftrightarrow & Q_1\varphi_2 =_E Q_2\varphi_2 \end{array}$$

d'où $\varphi_1 \approx_E \varphi_2$.
Réciproquement, supposons que $\varphi_1 \approx_E \varphi_2$. Soit Q_1, Q_2 deux termes tels que $\mathrm{var}(Q_1,Q_2) \subseteq \mathrm{dom}(\varphi_1) = \mathrm{dom}(\varphi_2)$, et $\mathrm{names}(Q_1,Q_2) \cap \mathrm{names}(\varphi_1^0,\varphi_2^0) = \emptyset$. Étant donné que pour $i,j \in \{1,2\}$ on a

$$\begin{array}{c} Q_j\varphi_i^0 =_E Q_j\{x \mapsto M\}\varphi_i \\ \mathrm{names}(Q_j\{x \mapsto M\}) \cap \mathrm{names}(\varphi_1,\varphi_2) = \emptyset \end{array}$$

on en déduit

$$\begin{array}{rc} & Q_1\varphi_1^0 =_E Q_2\varphi_1^0 \\ \Leftrightarrow & Q_1\{x \mapsto M\}\varphi_1 =_E Q_2\{x \mapsto M\}\varphi_1 \\ \Leftrightarrow & Q_1\{x \mapsto M\}\varphi_2 =_E Q_2\{x \mapsto M\}\varphi_2 \\ \Leftrightarrow & Q_1\varphi_2^0 =_E Q_2\varphi_2^0 \end{array}$$

– **Encryption Analysis-2**. Étant donné le test $\mathsf{pdec\_success}(\{x\}_{\vec{M}^{-1}}, N) =^? 1$ (respectivement $\mathsf{sdec\_success}(\{x\}_{\vec{M}^{-1}}, N) =^? 1$), vérifié dans φ_1^0 mais non dans φ_2^0, on a bien $\varphi_1^0 \not\approx_E \varphi_2^0$.

– **Encryption Analysis-1**. Nous considérons uniquement le cas du chiffrement asymétrique penc, le cas symétrique étant similaire.
Supposons que $\varphi_1^0 \approx_E \varphi_2^0$. Soit Q_1, Q_2 deux termes tels que $\mathrm{var}(Q_1,Q_2) \subseteq \mathrm{dom}(\varphi_1^1) = \mathrm{dom}(\varphi_2^1)$, et $\mathrm{names}(Q_1,Q_2) \cap \mathrm{names}(\varphi_1^0,\varphi_2^0) = \emptyset$. Étant donné que pour $i,j \in \{1,2\}$ on a

$$\begin{array}{c} Q_j\varphi_i^1 =_E Q_j\{y \mapsto \mathsf{pdec}(\{x\}_{\vec{M}^{-1}}, N)\}\,\varphi_i^0 \\ \mathrm{names}(Q_j\{y \mapsto \mathsf{pdec}(\{x\}_{\vec{M}^{-1}}, N)) \cap \mathrm{names}(\varphi_1^0,\varphi_2^0) = \emptyset \end{array}$$

on en déduit

$$\begin{array}{rc} & Q_1\varphi_1^1 =_E Q_2\varphi_1^1 \\ \Leftrightarrow & Q_1\{y \mapsto \mathsf{pdec}(\{x\}_{\vec{M}^{-1}}, N)\}\,\varphi_1^0 =_E Q_2\{y \mapsto \mathsf{pdec}(\{x\}_{\vec{M}^{-1}}, N)\}\,\varphi_1^0 \\ \Leftrightarrow & Q_1\{y \mapsto \mathsf{pdec}(\{x\}_{\vec{M}^{-1}}, N)\}\,\varphi_2 =_E Q_2\{y \mapsto \mathsf{pdec}(\{x\}_{\vec{M}^{-1}}, N)\}\,\varphi_2 \\ \Leftrightarrow & Q_1\varphi_2^1 =_E Q_2\varphi_2^1 \end{array}$$

Réciproquement, supposons que $\varphi_1^1 \approx_E \varphi_2^1$. Soit Q_1, Q_2 deux termes tels que $\mathrm{var}(Q_1, Q_2) \subseteq \mathrm{dom}(\varphi_1^0) = \mathrm{dom}(\varphi_2^0)$, et $\mathrm{names}(Q_1, Q_2) \cap \mathrm{names}(\varphi_1^0, \varphi_2^0) = \emptyset$. Soit r_0 un nom frais, et ρ_i le renommage $\rho_i = \{r_i \mapsto r_0\}$. Sachant que r_i n'apparaît nulle part ailleurs dans φ_i^0, on a pour tout $i, j \in \{1, 2\}$,

$$\begin{array}{c} Q_j\varphi_i^0\rho_i =_E Q_j\{x \mapsto \{\mathsf{penc}(y, \mathsf{pub}(N), r_0)\}_{\vec{M}}\}\,\varphi_i^1 \\ \mathrm{names}(Q_j\{x \mapsto \{\mathsf{penc}(y, \mathsf{pub}(N), r_0)\}_{\vec{M}}\}) \cap \mathrm{names}(\varphi_1^0, \varphi_2^0) = \emptyset \end{array}$$

Par conséquent,

$$\begin{array}{rcl} Q_1\varphi_1^0 & =_E & Q_2\varphi_1^0 \\ & \Leftrightarrow & \\ Q_1\varphi_1^0\rho_1 & =_E & Q_2\varphi_1^0\rho_1 \\ & \Leftrightarrow & \\ Q_1\{x \mapsto \{\mathsf{penc}(y, \mathsf{pub}(N), r_0)\}_{\vec{M}}\}\,\varphi_1^1 & =_E & Q_2\{x \mapsto \{\mathsf{penc}(y, \mathsf{pub}(N), r_0)\}_{\vec{M}}\}\,\varphi_1^1 \\ & \Leftrightarrow & \\ Q_1\{x \mapsto \{\mathsf{penc}(y, \mathsf{pub}(N), r_0)\}_{\vec{M}}\}\,\varphi_2^1 & =_E & Q_2\{x \mapsto \{\mathsf{penc}(y, \mathsf{pub}(N), r_0)\}_{\vec{M}}\}\,\varphi_2^1 \\ & \Leftrightarrow & \\ Q_1\varphi_2\rho_2 & =_E & Q_2\varphi_2\rho_2 \\ & \Leftrightarrow & \\ Q_1\varphi_2 & =_E & Q_2\varphi_2 \end{array}$$

– **Standardize**. Montrons que $\varphi_1^0 \approx_E \varphi_1^1$.
Soit M, N deux termes tels que $\mathrm{var}(M, N) \subseteq \mathrm{dom}(\varphi_i^j)$ et $\mathrm{names}(M, N) \cap \mathrm{names}(\varphi_1^0, \varphi_1^1) = \emptyset$. Si $M\varphi_1^0 =_E N\varphi_1^0$, alors

$$\begin{array}{rcl} M\varphi_1^1 & =_E & (M\varphi_1^0)\{a \mapsto \{a'\}_{\vec{C}[a_1,\ldots,a_n]^{-1}}\} \\ & =_E & (N\varphi_1^0)\{a \mapsto \{a'\}_{\vec{C}[a_1,\ldots,a_n]^{-1}}\} \\ & =_E & N\varphi_1^1 \end{array}$$

Réciproquement, si $M\varphi_1^1 =_E N\varphi_1^1$, alors

$$\begin{array}{rcl} M\varphi_1^0 & =_E & (M\varphi_1^1)\{a' \mapsto \{a\}_{\vec{C}[a_1,\ldots,a_n]}\} \\ & =_E & (N\varphi_1^1)\{a' \mapsto \{a\}_{\vec{C}[a_1,\ldots,a_n]}\} \\ & =_E & N\varphi_1^0 \end{array}$$

– **Solve**. Évident.

6.4.6 Sûreté calculatoire

Nous commençons par prouver deux lemmes techniques. Le premier justifie le fait que les distributions aléatoires ($\xleftarrow{R} [\![\tau]\!]_\eta$) construites à la section 6.3.2 sont sans collisions.

Lemme 6.15. *Supposons l'implémentation sûre (au sens précédent). Alors, pour tout type τ, la probabilité de collision relative à la distribution ($\xleftarrow{R} [\![\tau]\!]_\eta$) est une fonction négligeable du paramètre de complexité η.*

Démonstration. Nous établissons cette propriété par récurrence sur τ.

Les cas des types $\tau = Data$, $\tau = Coins$ sont clairs dans la mesure où $\alpha_1(\eta)$ et $\alpha_2(\eta)$ croissent au moins linéairement. Le cas $\tau = Pair[\tau_1, \tau_2]$ est clair également par l'hypothèse d'induction sur τ_1 et τ_2.

Pour $\tau \in \{SKey, DKey, EKey\}$, si la distribution ($\xleftarrow{R} [\![\tau]\!]$) a une probabilité de collision non négligeable, alors en utilisant l'algorithme de génération de clé, un adversaire obtient avec une probabilité non négligeable la clé secrète de l'expérience de T_{senc}-sécurité ou, respectivement, une clé de déchiffrement convenable dans l'expérience de T_{penc}-sécurité. Il est alors facile de contredire ces deux critères.

Finalement, supposons que $\tau = SCipher[\tau_1]$ (le cas $\tau = ACipher[\tau_1]$ est similaire), et que, par contradiction, ($\xleftarrow{R} [\![\tau]\!]_\eta$) n'est pas sans collisions, c'est-à-dire que :

$$\mathbb{P}\Big[e_1, e_2 \xleftarrow{R} [\![\tau_1]\!]_\eta; k_1, k_2 \xleftarrow{R} [\![SKey]\!]_\eta; r_1, r_2 \xleftarrow{R} [\![Coins]\!]_\eta \ : \ \mathcal{E}^s(e_1, k_1, r_1) = \mathcal{E}^s(e_2, k_2, r_2)\Big]$$

n'est pas négligeable. Étant donné l'équation du déchiffrement, ceci implique que la probabilité suivante est également non négligeable :

$$\mathbb{P}\Big[e_1, e_2 \xleftarrow{R} [\![\tau_1]\!]_\eta; k_1, k_2 \xleftarrow{R} [\![SKey]\!]_\eta; r_2 \xleftarrow{R} [\![Coins]\!]_\eta \ : \ e_1 = \mathcal{D}^s(\mathcal{E}^s(e_2, k_2, r_2), k_1)\Big]$$

Comme le membre de droite du test d'égalité ne dépend pas de e_1, on en déduit en particulier que ($\xleftarrow{R} [\![\tau_1]\!]_\eta$) n'est pas sans collisions, ce qui contredit l'hypothèse d'induction. □

Notre second lemme est un renforcement du critère de T_{enc}-Pwd-sécurité (définition 6.3).

Lemme 6.16. *Soit $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ un procédé de chiffrement déterministe surjectif vérifiant le critère de sécurité T_{enc}-Pwd.*

Étant donné un paramètre η, $k_1, \ldots, k_n \in \{0,1\}^{\alpha_1(\eta)}$, et $\ell_1, \ldots, \ell_n \in \{+1, -1\}$, on appelle clé vectorielle *l'expression formelle $\vec{k} = k_1^{\ell_1} \cdots k_n^{\ell_n}$. L'ensemble de telles expressions pour un η fixé est noté $\mathcal{G}_\eta$. Pour tout $\vec{k} \in \mathcal{G}_\eta$, on pose $|\vec{k}| = \sum_i |\ell_i| = n$. On écrit également $\mathcal{E}(e, \vec{k})$ le résultat de la séquence de chiffrements et de déchiffrements par $\vec{k}$ sur la chaîne e ($e \in [\![\tau]\!]_\eta$ pour un certain $\tau \in T_{\mathsf{enc}}$), par analogie avec le chiffrement symbolique.*

Soit M un entier positif. Pour tout paramètre de sécurité η, pour tout type $\tau \in T_{\mathsf{enc}}$, considérons l'expérience suivante engageant un adversaire PPTIME en deux étapes $A = (A_1, A_2)$:

- *A_1 choisit et envoie une clé vectorielle $\vec{k} \in \mathcal{G}_\eta$ telle que $|\vec{k}| \leq M$ ainsi qu'une information d'état st ;*
- *un bit $b \xleftarrow{R} \{0,1\}$ est choisi aléatoirement ; si $b = 0$, on pose $m \xleftarrow{R} [\![\tau]\!]_\eta$ et $c = \mathcal{E}(m, \vec{k})$; sinon, $c \xleftarrow{R} [\![\tau]\!]_\eta$ est pris aléatoirement dans $[\![\tau]\!]_\eta$;*
- *A_2 reçoit c et st, et répond par un bit b'.*

L'adversaire A gagne ssi $b' = b$.

Alors, l'avantage de A, défini par $\mathrm{Adv}^{\tau,M}_{Pwd,\Pi,A}(\eta) = \mathbb{P}[A \text{ gagne}] - \frac{1}{2}$, est négligeable.

Démonstration. Fixons M et τ. Pour tout $0 \leq N \leq M$, on considère l'expérience « hybride » suivante sur $A = (A_1, A_2)$:

- A_1 choisit et envoie une clé vectorielle $\vec{k} = k_1^{\ell_1} \cdots k_n^{\ell_n} \in \mathcal{G}_\eta$ telle que $|\vec{k}| = n \leq M$ ainsi qu'une information d'état st ;
- soit $\vec{k}_N = \begin{cases} \epsilon & \text{si } n \leq N \\ k_{N+1}^{\ell_{N+1}} \cdots k_n^{\ell_n} & \text{sinon} \end{cases}$
- soit $b \xleftarrow{R} \{0,1\}$ et $m \xleftarrow{R} [\![\tau]\!]_\eta$;
- on pose $c = \mathcal{E}(m, \vec{k})$ si $b = 0$, et $c = \mathcal{E}(m, \vec{k}_N)$ dans le cas contraire ;
- A_2 reçoit c et st, et répond par un bit b'.

L'adversaire A gagne ssi $b' = b$. Notons $\mathrm{Adv}^N_A(\eta) = \mathbb{P}[A \text{ gagne}] - \frac{1}{2}$ son avantage.

Le but de A dans cette expérience est donc de deviner si l'environnement a ignoré ou non les N premiers étages de la séquence de chiffrements/déchiffrements supposée être appliquée à un nombre aléatoire.

Pour $N = 0$, la valeur c suit la même distribution quelle que soit la valeur de b ; par conséquent pour tout A, $\mathrm{Adv}^0_A(\eta) = 0$.

Pour $N = M$, les deux distributions possibles de c sont les mêmes que dans l'expérience de l'énoncé. Par conséquent, $\mathrm{Adv}^M_A(\eta) = \mathrm{Adv}^{\tau,M}_{\mathrm{Pwd},\Pi,A}(\eta)$.

Montrons que pour tout adversaire A et pour tout $0 \leq N < M$, $|\mathrm{Adv}^N_A(\eta) - \mathrm{Adv}^{N+1}_A(\eta)| \leq 2|\mathrm{Adv}^{\tau,M}_{\mathrm{Pwd},\Pi,B^N}(\eta)|$ pour un certain adversaire polynomial B^N contre le jeu de la T_{enc}-sécurité.

En effet, on construit $B^N = (B_1, B_2)$ à partir de $A = (A_1, A_2)$ de la manière suivante :

- B_1 : soit $\vec{k}$ et st les sorties de A_1 ; soit $\vec{k}_N$ et $\vec{k}_{N+1}$ les vecteurs construits comme ci-dessus à partir de $\vec{k}$; par construction, trois cas sont possibles :
  (i) $\vec{k}_N = \vec{k}_{N+1}$,
  (ii) $\vec{k}_N = k_0 \cdot \vec{k}_{N+1}$ pour un certain k_0, ou encore
  (iii) $\vec{k}_N = k_0^{-1} \cdot \vec{k}_{N+1}$ pour un certain k_0 ;
  dans le cas (i), B_1 renvoie $k_0 = 0^\eta$ et $st' = (st, 0, \epsilon)$; dans le cas (ii), B_1 renvoie k_0 et $st' = (st, 1, \vec{k}_{N+1})$; dans le cas (iii), B_1 renvoie k_0 et $st' = (st, 1, \vec{k}_N)$;
- B_2 : soit c et $st' = (st, i, k_0, \vec{k}')$ la réponse de l'environnement ; si $i = 0$, B_2 renvoie un bit b' tiré au sort ; sinon, B_2 interroge l'attaquant A_2 sur l'entrée st et $c' = \mathcal{E}(c, \vec{k}')$, et renvoie la même réponse.

On remarque que dans les cas (ii) et (iii), A reçoit les entrées du cas $b = 1$ de l'expérience hybride au niveau N ou au niveau $N + 1$ — selon que le c renvoyé ci-dessus est un nombre aléatoire ou le chiffré par k_0 d'un nombre aléatoire. Comme le cas (i) et le cas $b = 0$ ne contribuent pas à la différence d'avantage de A entre les niveaux N et $N + 1$, on en déduit l'inégalité souhaitée par le lemme de Shoup [Sho04] : pour tous événements E, E', F, si $\mathbb{P}[E \wedge \neg F] = \mathbb{P}[E' \wedge \neg F]$ alors $|\mathbb{P}[E] - \mathbb{P}[E']| \leq \mathbb{P}[F]$.

Dans la mesure où chaque $|\operatorname{Adv}^{\tau,M}_{\mathrm{Pwd},\Pi,B^N}(\eta)|$ est négligeable par hypothèse, on obtient finalement par inégalité triangulaire que l'avantage $\operatorname{Adv}^{\tau,M}_{\mathrm{Pwd},\Pi,A}(\eta)$ est négligeable. □

Nous prouvons maintenant la sûreté calculatoire de chacune des règles de transformation. Comme précédemment, on écrit $\varepsilon = \varphi_1^0 \approx^? \varphi_2^0$ pour les équations de départ de règles et $\varepsilon' = \varphi_1^1 \approx^? \varphi_2^1$ pour les équations d'arrivée.

– **Undecipherable Encryption**. Pour tout terme T, soit $|T|_e$ le nombre de sous-terme distincts dans T de la forme $\mathsf{pub}(U)$, $\mathsf{enc}(U, V)$, $\mathsf{penc}(U, V, W)$, ou $\mathsf{senc}(U, V, W)$. Cette notation est étendue de la manière attendue aux n-uplets de termes et aux environnements.
Étant donné un type τ, $\mathrm{PPos}(\tau)$ désigne l'ensemble des positions p dans τ telles que pour tout préfixe strict q de p, le symbole en position q dans τ soit $\tau(q) = \mathit{Pair}$. Pour tout terme de type τ et pour tout $p \in \mathrm{PPos}(\tau)$, on note $\pi_p(T)$ le terme de sorte $\tau|_p$ défini par

$$\begin{aligned} \pi_\epsilon(T) &= T \\ \pi_{1\cdot q}(T) &= \mathsf{fst}(\pi_q(T)) \\ \pi_{2\cdot q}(T) &= \mathsf{snd}(\pi_q(T)) \end{aligned}$$

Nous prouvons les propriétés suivantes par récurrence sur $N \geq 0$:
– $P(N)$—Dans les conditions d'application de **Undecipherable Encryption**, c'est-à-dire lorsque :
  – φ_1 est un environnement bien formé ;
  – k est un nom dont toutes les occurrences dans φ sont en position de clé ;
  – T est un sous-terme maximal de φ_1 de la forme $\mathsf{pub}(k)$, $\mathsf{penc}(U, k', r)$, $\mathsf{senc}(U, k, r)$ ou $\mathsf{enc}(U, k)$;
  – n est un nom frais ;
  si de plus $|\varphi_1|_e \leq N$, alors $[\![\varphi_1]\!] \approx [\![\varphi_1\{T \mapsto n\}]\!]$;
– $Q(N)$—Pour tout environnement bien formé $\varphi = \{x_1 \mapsto T_1, x_2 \mapsto T_2\}$ tel que $|\varphi|_e \leq N$, si T_1 et T_2 sont du même type τ, ne contiennent aucune des constantes $\mathsf{w}, \mathsf{c}_0, \mathsf{c}_1 \ldots$, et n'ont pas de sous-termes de la forme $\mathsf{enc}(S, 0)$ ou $\mathsf{enc}(S, 1)$, alors pour tout $p \in \mathrm{PPos}(\tau)$, $\pi_p(T_1) \neq_E \pi_p(T_2)$ implique que la quantité

$$\mathbb{P}\left[e_1, e_2 \xleftarrow{R} [\![\pi_p(T_1), \pi_p(T_2)]\!]_\eta \ : \ e_1 = e_2\right]$$

est une fonction négligeable de η.

Plus précisément, nous prouvons les quatre énoncés suivants :

(1) $P(0)$,

(2) $P(N + 1) \Leftarrow Q(N)$,

(3) $Q(0)$,

(4) $Q(N+1) \Leftarrow (P(N+1) \text{ et } Q(N))$.

(1) $P(0)$ est vrai trivialement.

(2) $P(N+1) \Leftarrow Q(N)$. Soit $\varphi_1 = \{x_1 \mapsto T_1, \ldots, x_m \mapsto T_m\}$, et $\varphi_1' = \varphi_1\{T \mapsto n\}$. Nous distinguons différents cas suivant la forme de T.

– Si $T = \mathsf{pub}(k)$, nous avons $[\![\varphi_1]\!] = [\![\varphi_1']\!]$ puisque par hypothèse k apparaît seulement en position de clé, autrement dit, vu le système de types, sous le symbole pub.

– Si $T = \mathsf{enc}(U, k)$, alors soit τ_0 la sorte de U. Étant donné un adversaire A capable de distinguer efficacement $[\![\varphi_1]\!]$ de $[\![\varphi_1']\!]$, nous construisons un adversaire B contre la T_{enc}-sécurité de la manière suivante :

1. pour chaque nom $a \neq k$ de sorte τ apparaissant dans φ_1, tirer une valeur $\widehat{a} \xleftarrow{R} [\![s]\!]_\eta$ au sort ;
2. pour chaque x_i $(1 \leq i \leq m)$ de sorte τ_i, calculer $\widehat{T_i} \in [\![\tau_i]\!]_\eta$ récursivement comme suit :

$$\begin{aligned}
\widehat{\mathsf{enc}_\tau(T,k)} &= \mathcal{E}(\widehat{T}) \text{ si } T \neq U \\
\widehat{\mathsf{enc}_{\tau_0}(U,k)} &= \mathcal{E}^*(\widehat{U}) \\
\widehat{f(T_1,\ldots,T_n)} &= [\![f]\!]_\eta(\widehat{T_1},\ldots,\widehat{T_n}) \\
&\qquad \text{si } f(T_1,\ldots,T_n) \notin \{\mathsf{enc}(T',k), \mathsf{dec}(T',k)\}
\end{aligned}$$

   où $\mathcal{E}(.)$ désigne l'oracle de chiffrement pour le jeu de T_{enc}-sécurité, et $\mathcal{E}^*(\widehat{U})$ désigne la réponse à la *requête-défi* $\widehat{U}$ (*challenge ciphertext*).
3. soumettre l'environnement concret $\{x_1 \mapsto \widehat{T_1}, \ldots, x_n \mapsto \widehat{T_n}\}$ à A et renvoyer le même résultat.

Notons que par maximalité T n'est pas sous-terme d'un chiffrement par k, ainsi nous pouvons supposer que B envoie de manière correcte la requête-défi en dernier. Nous avons également utilisé le fait que k (dont la valeur concrète n'est pas connu par B pendant l'expérience) apparaît uniquement en position de clé de chiffrement dans φ_1.
La distribution résultant du calcul de B et soumise à A vaut ou bien $[\![\varphi_1]\!]_\eta$ ou bien $[\![\varphi_1']\!]_\eta$ selon la valeur de $\mathcal{E}^*(\widehat{U})$, à savoir respectivement, le chiffrement de $\widehat{U}$ ou une valeur aléatoire dans $[\![\tau]\!]_\eta$. Ainsi la probabilité que B devine la réponse correcte est la même que pour A. Néanmoins, il se peut que B ne remplisse pas la condition supplémentaire pour le gain du jeu de T_{enc}-sécurité, c'est-à-dire : ne pas soumettre à l'oracle de chiffrement un message en clair déjà soumis comme requête-défi. Ceci peut se produire s'il existe un sous-terme $\mathsf{enc}_{\tau_0}(U', k)$ tel que U et U' ont le même type, $U' \neq U$ et $\widehat{U'} = \widehat{U}$.
Supposons que ce soit le cas. Soit $\varphi = \{x \mapsto U, y \mapsto U'\}$. Comme U' et $T = \mathsf{enc}_{\tau_0}(U, k)$ sont deux sous-termes de φ_1, et que T n'est pas un sous-terme de U' (par maximalité), on a $|\varphi|_e < |\varphi_1|_e = N + 1$. De plus, comme φ_1 est bien formé, c'est le cas de φ, et φ ne contient aucun sous-terme $\mathsf{enc}(T', c)$. Par conséquent, l'hypothèse d'induction $Q(N)$ implique que la probabilité que $\widehat{U'} = \widehat{U}$ est négligeable.

– Si T est de la forme $T = \mathsf{penc}(U, k', r)$ ou $T = \mathsf{senc}(U, k, r)$, étant donné un adversaire A capable de distinguer $[\![\varphi_1]\!]$ et $[\![\varphi'_1]\!]$, on construit un adversaire B contre la T_{penc}-sécurité (ou respectivement contre la T_{senc}-sécurité) de la même manière que ci-dessus. On utilise cette fois le fait que les jetons r apparaissent dans au plus un terme de chiffrement. Notons qu'il n'y a pas de condition supplémentaire à vérifier dans ces deux cas.

Avant de prouver (3) et (4), notons que $Q(N)$ reste inchangé si nous restreignons les valeurs de p aux positions les plus basses dans $\mathrm{PPos}(\tau)$, autrement dit, celles pour lesquelles $\tau(p) \neq \mathit{Pair}$. En effet, si T_1 et T_2 sont de type $\mathit{Pair}[\tau_1, \tau_2]$, d'une part, du côté symbolique, on a

$$T_1 =_E T_2 \text{ ssi } \mathsf{fst}(T_1) =_E \mathsf{fst}(T_2) \text{ et } \mathsf{snd}(T_1) =_E \mathsf{snd}(T_2)$$

et d'autre part, du côté cryptographique, la probabilité de collision

$$\mathbb{P}\left[e_1, e_2 \leftarrow [\![T_1, T_2]\!]_\eta \ : \ e_1 = e_2\right]$$

est négligeable ssi

$$\begin{aligned} & \mathbb{P}\left[e_1, e_2 \leftarrow [\![\mathsf{fst}(T_1), \mathsf{fst}(T_2)]\!]_\eta \ : \ e_1 = e_2\right] \\ \text{et} \quad & \mathbb{P}\left[e_1, e_2 \leftarrow [\![\mathsf{snd}(T_1), \mathsf{snd}(T_2)]\!]_\eta \ : \ e_1 = e_2\right] \end{aligned}$$

est négligeable.
Nous montrons maintenant la propriété (4). (La propriété (3) se traite de manière identique.) Soit $\varphi = \{x_1 \mapsto T_1,\ x_2 \mapsto T_2\}$ un environnement bien formé tel que $|\varphi|_e \leq N + 1$ et que T_1 et T_2 ont un même type τ_0, ne contiennent pas de constantes $\mathsf{w}, \mathsf{c}_0, \mathsf{c}_1, \ldots$, ni de sous-terme de la forme $\mathsf{enc}(S, 0)$ ou $\mathsf{enc}(S, 1)$. Soit $p_0 \in \mathrm{PPos}(\tau_0)$ et supposons $\pi_{p_0}(T_1) \neq_E \pi_{p_0}(T_2)$.
Soit $T'_1 = \pi_{p_0}(T_1)\!\downarrow_{\mathcal{R}}$, $T'_2 = \pi_{p_0}(T_2)\!\downarrow_{\mathcal{R}}$ et soit τ la sorte de T'_1 et T'_2. D'après la discussion précédente, on peut supposer sans perte de généralité que p_0 est maximal dans $\mathrm{PPos}(\tau_0)$, *i.e.* τ n'est pas un type pair. Par conséquent, τ est d'une des formes suivantes : SKey, EKey, DKey, Data, Coins, $\mathit{SCipher}[\tau']$, ou $\mathit{ACipher}[\tau']$.
Étant donné que T'_1 et T'_2 sont $\mathcal{R}$-réduits, que φ est bien formé, que T'_1 et T'_2 (en tant que sous-termes de T_1 et T_2 respectivement) ne contiennent pas de constantes $\mathsf{w}, \mathsf{c}_0, \mathsf{c}_1, \ldots$ ni de sous-terme de la forme $\mathsf{enc}(S, 0)$ ou $\mathsf{enc}(S, 1)$, et que $T_{\mathsf{enc}} \cap \{\mathit{Pair}[\tau_1, \tau_2]\}_{\tau_1, \tau_2} = \emptyset$, on en déduit par typage que T'_1 et T'_2 sont chacun d'une des formes suivantes :

(i) projection d'un nom $\pi_p(a)$, (éventuellement $p = \epsilon$)
(ii) constante 0 ou 1,
(iii) clé publique $\mathsf{pub}(k)$,
(iv) message chiffré déterministe $\mathsf{enc}(U, k)$,
(v) message chiffré probabiliste, $\mathsf{penc}(U, k', r)$, $\mathsf{penc}(U, \mathsf{pub}(k), r)$, ou encore $\mathsf{senc}(U, k, r)$.

Nous distinguons plusieurs cas selon la forme de T'_1 et T'_2.
– Si T'_1 et T'_2 sont tous deux des formes (i)–(iii), alors le résultat découle du lemme 6.15 concernant l'absence de collision dans les tirages aléatoires.
– Si l'un des deux termes exactement est de la forme (i)–(iii), par exemple T'_1, alors on distingue à nouveau deux cas :

– Si T'_2 est un message chiffré de la forme $\mathsf{enc}(U,k)$, $\mathsf{penc}(U,\mathsf{pub}(k),r)$, ou $\mathsf{senc}(U,k,r)$, et que $T'_1 = k$, alors la probabilité

$$\mathbb{P}\left[e_1, e_2 \xleftarrow{R} [\![T'_1, T'_2]\!]_\eta \ : \ e_1 = e_2\right]$$

est négligeable, sans quoi la sécurité du chiffrement serait compromise car l'on pourrait facilement récupérer la clé k pendant l'expérience. (On calcule la valeur concrète de T'_2 comme précédemment, sachant que k apparaît seulement en position de clé dans U.)

– Dans le cas contraire, au choix

(i) $T'_2 = \mathsf{penc}(U,k',r)$, ou bien

(ii) T'_2 est de la forme $\mathsf{enc}(U,k)$, $\mathsf{penc}(U,\mathsf{pub}(k),r)$, ou $\mathsf{senc}(U,k,r)$, tandis que $T'_1 = \mathsf{pub}(k)$ ou bien k n'apparaît nulle part dans T'_1.

Si la probabilité de collision ci-dessus n'est pas négligeable, alors calculer T'_1 permet de prévoir la valeur de T'_2 avec une chance de succès non négligeable. À nouveau ceci contredit la sécurité du chiffrement de T'_2.

– Finalement, si T'_1 et T'_2 sont tous les deux de la forme (iv) ou (v), alors comme φ est bien formé, on peut supposer par exemple que T'_1 est de la forme $\mathsf{enc}(U,k)$, $\mathsf{penc}(U,\mathsf{pub}(k),r)$, ou $\mathsf{senc}(U,k,r)$ (selon la sorte de k), tandis que la clé k apparaît seulement en position de clé dans T'_2 et U. Soit T un sous-terme maximal de T'_1 et T'_2 de la forme $\mathsf{enc}(U',k)$, $\mathsf{penc}(U',\mathsf{pub}(k),r')$, ou $\mathsf{senc}(U',k,r')$.
Soit n un nom frais, et y_1, y_2 deux variables de même sorte que T'_1. Soit $\varphi' = \{y_1 \mapsto T'_1,\ y_2 \mapsto T'_2\}$. Comme φ est bien formé, et sachant la forme de T'_1 et T'_2, φ' et T satisfont les conditions de $P(N+1)$. On en déduit que la probabilité de collision entre T'_1 et T'_2 est égale à celle entre $T''_1 = T'_1\{T \mapsto n\}$ et $T''_2 = T'_2\{T \mapsto n\}$ à une différence négligeable près. Nous concluons en appliquant $Q(N)$ à l'environnement résultant $\varphi'' = \{y_1 \mapsto T''_1,\ y_2 \mapsto T''_2\}$.

Ceci conclue la preuve de $P(N)$ et de $Q(N)$ pour tout N. En particulier, dans les conditions d'application de la règle **Undecipherable Encryption**, on a bien $[\![\varphi_1^0]\!] \approx [\![\varphi_1^1]\!]$

– **Split Names**. On a $[\![\varphi_1^0]\!] = [\![\varphi_1^1]\!]$ par définition du tirage aléatoires des valeurs de type $Pair[\tau_1, \tau_2]$.

– **Pair Analysis**. Étant donné la sémantique des paires et le lemme 5.1 sur la correction calculatoire des égalités symboliques, on construit facilement un adversaire (efficace) contre l'indistinguabilité $[\![\varphi_1^1]\!] \approx [\![\varphi_2^1]\!]$ à partir d'un adversaire (efficace) contre $[\![\varphi_1^0]\!] \approx [\![\varphi_2^0]\!]$.

– **Redundancy Analysis-2**. Trivial.

– **Redundancy Analysis-1**. Étant donné un adversaire A contre $[\![\varphi_1^0]\!] \approx [\![\varphi_2^0]\!]$, nous construisons un adversaire B contre $[\![\varphi_1]\!] \approx [\![\varphi_2]\!]$. En effet, la relation $x\varphi_i^0 =_E M\varphi_i$ est vérifiée concrètement par le lemme 5.1. Étant donné un tirage ϕ de l'environnement φ_i, B interroge A avec $\phi^0 = \phi \uplus \{x \mapsto [\![M]\!]_{\eta,\phi}\}$, et renvoie le même résultat que A.

– **Encryption Analysis-2**. Trivial.

– **Encryption Analysis-1**. Nous considérons uniquement le cas du chiffrement asymétrique, le cas symétrique étant identique.
Étant donné un adversaire A contre $[\![\varphi_1^0]\!] \approx [\![\varphi_2^0]\!]$, nous construisons un adversaire B contre $[\![\varphi_1^1]\!] \approx [\![\varphi_2^1]\!]$ de la manière suivante. Soit r_0 un nom frais de sorte *Coins* et ρ_i le renommage $\rho_i = \{r_i \mapsto r_0\}$ comme précédemment. Par le lemme 5.1, la relation symbolique

$$x\varphi_i^0\rho_i =_E \{\mathsf{penc}(y, \mathsf{pub}(N), r_0)\}_{\vec{M}}\varphi_i^1$$

(qui repose sur la condition sur les jetons r_i) est vérifiée également au niveau concret. φ_i^0 et $\varphi_i^0\rho_i$ ont clairement la même sémantique. Étant donné un tirage ϕ^1 de φ_i^1, B interroge A avec

$$\phi^0 = \phi^1 \uplus \{x \mapsto [\![\{\mathsf{penc}(y, \mathsf{pub}(N), r_0)\}_{\vec{M}}]\!]_{\eta,\phi^1}\}|_{\mathrm{dom}(\varphi^0)},$$

et renvoie le même résultat que A.

– **Standardize**. Montrons que $[\![\varphi_1^0]\!] \approx [\![\varphi_1^1]\!]$. En effet, on a $\varphi_1^1 =_E \varphi_1^0\{a \mapsto \{a'\}_{\vec{C}[a_1,\ldots,a_n]^{-1}}\}$ où a' est un nom frais. Soit τ_i la sorte de a_i. Par le lemme 5.1, ceci implique

$$\begin{aligned}[\![\varphi_1^1]\!]_\eta &= [\![\varphi_1^0\{a \mapsto \{a'\}_{\vec{C}[a_1,\ldots,a_n]^{-1}}\}]\!]_\eta \\ &= [\![\varphi_1^0]\!]_{\eta\,,\, e_i \xleftarrow{R} [\![\tau_i]\!]\,;\, a_i \mapsto e_i, a \mapsto [\![\{a'\}_{\vec{C}[a_1,\ldots,a_n]^{-1}}]\!]_{\eta, a_i \mapsto e_i}}\end{aligned}$$

Le lemme 6.16 implique donc $[\![\varphi_1^0]\!] \approx [\![\varphi_1^1]\!]$.

– **Solve**. Supposons que $\varepsilon \neq \bot$, c'est-à-dire $\varphi_1 \approx_E \varphi_2$. Par définition de la sémantique idéale des environnements (chapitre 5), on a donc $[\![\varphi_1]\!]^{ideal} = [\![\varphi_2]\!]^{ideal}$.
Or, sachant que l'implémentation est inconditionnellement $=_E$-sûre (par le lemme 5.1) et que pour $i \in \{1, 2\}$, l'environnement φ_i est directement transparent, par la proposition 5.17, on a

$$[\![\varphi_i]\!] = [\![\varphi_i]\!]^{ideal}$$

D'où, finalement $[\![\varphi_1]\!] = [\![\varphi_2]\!]$.

6.5 Comparaisons

Les résultats décrits dans les deux précédents chapitres — obtenus en collaboration avec Véronique Cortier, Steve Kremer [BCK05], Martín Abadi et Bogdan Warinschi [ABW06] — constituent à notre connaissance la première justification cryptographique de l'équivalence statique, en l'occurrence pour les théories équationnelles du OU exclusif (pure) et du chiffrement (symétrique, asymétrique et symétrique déterministe surjectif avec clés faibles).[1]

Parmi les travaux existants sur le OU exclusif, M. Backes et B. Pfitzmann [BP05] ont montré récemment un résultat d'impossibilité dans le formalisme de la simulabilité (*simulatability*) en présence d'un intrus actif et d'autres opérations cryptographiques usuelles. Ils présentent également un résultat de sûreté dans le même cadre,

[1] L'article d'ICALP'05 [BCK05] comporte un exemple supplémentaire que nous n'avons pas reproduit ici : la théorie du chiffrement déterministe surjectif et des listes.

restreint à un intrus passif. Ce dernier résultat ne semble toutefois pas englober les propriétés d'indistinguabilité au sens de l'équivalence statique.

P. Laud présente également deux résultats liés au OU exclusif et au chiffrement symétrique déterministe dans son mémoire de thèse [Lau02]. D'une part, il propose un système de preuve (manuelle) prenant en compte le chiffrement symétrique déterministe (éventuellement surjectif) et le OU exclusif dans le cas passif.

D'autre part, P. Laud montre la sûreté cryptographique d'un critère formel d'équivalence à base de motifs (*patterns*) dans la lignée de M. Abadi et de P. Rogaway [AR02]. Le critère s'applique à un chiffrement non nécessairement surjectif si bien que certaines équivalences prouvables dans notre formalisme ne le sont pas selon le critère de P. Laud. Par exemple, nous pouvons considérer que le chiffrement d'un nombre aléatoire frais par une clé connue est indistinguable d'un nombre aléatoire frais :

$$\{x \mapsto \mathsf{enc}(n, k),\ y \mapsto k\} \approx_E \{x \mapsto n',\ y \mapsto k\}$$

Du point de vue formel, cette équivalence découle de la définition de l'équivalence statique et de la présence de l'équation $\mathsf{enc}(\mathsf{dec}(x, y), y) = x$.

Définir dans un cadre non équationnel un critère formel incluant des équivalences comme celle ci-dessus ne nous paraît pas immédiat. Toutefois, M. Abadi et B. Warinchi [AW05] proposent un critère formel de résistance aux attaques par dictionnaire dans le cas passif, dont la définition, non équationnelle, repose sur des motifs (*patterns*) [AR02]. Au prix d'une notion d'équivalence plus abstraite — à savoir l'équivalence statique —, la seconde application de ce chapitre constitue une généralisation de ce résultat à des équivalences quelconques entre environnements.

Chapitre 7

Conclusion

D'une manière générale, étudier la sécurité d'un protocole cryptographique suppose le choix d'un modèle de sécurité.

Les modèles calculatoires (ou cryptographiques) procurent les meilleures garanties de sécurité, mais nécessitent a priori une preuve manuelle pour chaque protocole.

À l'inverse, les modèles symboliques (ou logiques) permettent d'analyser les protocoles de manière automatisée mais ne prennent pas en compte la totalité des attaques cryptographiques. De plus, certaines propriétés et certains protocoles peuvent s'avérer difficiles à modéliser et à vérifier dans un cadre symbolique.

Partant de ce constat, nous avons abordé dans cette thèse deux axes de recherche importants de la vérification des protocoles cryptographiques, à savoir, d'une manière générale :

- l'extension des méthodes d'analyse logique à de nouvelles propriétés de sécurité des protocoles, et
- la justification calculatoire des modèles symboliques.

Modélisation et vérification des propriétés d'équivalence

De nombreuses propriétés de sécurité, en particulier dans le modèle calculatoire, s'énoncent naturellement en terme d'indistinguabilité. Modéliser ce type de propriétés dans un cadre logique est plus complexe et repose généralement sur une notion d'équivalence observationnelle, définie dans un langage adapté.

Dans le chapitre 3, nous avons ainsi proposé un langage de spécification inspiré du pi-calcul appliqué [AF01], et permettant de spécifier les propriétés de sécurité les plus courantes (secret simple, authentification) ainsi que des propriétés d'équivalence comme le secret fort et la résistance aux attaques par dictionnaire.

Pour montrer la décidabilité de ces propriétés (à l'exception du secret fort) dans le cas d'un nombre borné de sessions, nous avons procédé en deux étapes.

Tout d'abord (chapitre 3), nous avons muni le langage d'une sémantique symbolique correcte et complète vis à vis des propriétés étudiées, ce pour toute théorie équationnelle. À notre connaissance, cette sémantique symbolique à base de biprocessus constitue un résultat original, dans la mesure où les travaux existants dans ce domaine sont généralement restreints à un jeu de primitives fixés [AG97, BN05, DSV03, BdNP99, Cor02], aux propriétés d'accessibilité [BB05, ZD06] ou encore ne comportent pas de sémantique symbolique [AF01, BAF05].

Dans un second temps (chapitre 4), nous avons présenté une procédure de décision pour les systèmes de contraintes engendrés par la sémantique symbolique, dans le cas des théories équationnelles sous-terme-convergentes et des processus positifs (*i.e.* sans test de différence). Du fait des propriétés d'équivalence en vue, il s'agissait de résoudre non seulement la satisfiabilité de ces systèmes de contraintes, mais aussi leur (S-)équivalence.

Ce résultat constitue ainsi une généralisation à la fois des premiers travaux de M. Abadi et V. Cortier [AC04] sur l'équivalence statique, et des travaux de S. Delaune et F. Jacquemard [DJ04a] sur la satisfiabilité des systèmes de contraintes de l'intrus. Dans le cas d'un nombre borné de sessions, il nous permet finalement d'obtenir la décidabilité, dans la classe co-NP, des différents problèmes de sécurité considérés, et notamment de la résistance aux attaques par dictionnaire.

Grâce au découpage de la preuve en deux étapes, la première étant valable pour toute théorie équationnelle, il est permis d'envisager des extensions de ce schéma à d'autres théories : il « suffit » en effet pour cela d'étudier les systèmes de contraintes des théories équationnelles considérées. Par ailleurs, deux questions ouvertes intéressantes restent la décidabilité de l'équivalence (pure) des systèmes de contraintes et la décidabilité de la S-équivalence en présence de tests de différence.

Enfin, une voie prometteuse abordée récemment par S. Delaune, S. Kremer et M. Ryan [Kre06] consiste à utiliser le résultat de décision de la S-équivalence pour décider des équivalences de processus plus expressives — en l'occurrence la may-équivalence et, de façon approchée, la congruence barbue. À moyen terme, il est également permis d'imaginer des extensions de ce schéma aux propriétés de jeux impliquées dans les protocoles de signature de contrat, ou encore à l'étude des protocoles de vote électronique.

Justification calculatoire de l'équivalence statique

La question de la sûreté calculatoire des outils d'analyse logique existants est intéressante et naturelle à plusieurs titres : d'un point de vue pratique tout d'abord, car une réponse positive fournit immédiatement une méthode de preuve automatique *et* sûre calculatoirement ; d'un point de vue théorique ensuite, car l'adéquation entre sécurité logique et sécurité prouvée cryptographique conforte à la fois les modèles logiques (comme étant suffisamment sûrs), et les modèles calculatoires (comme étant suffisamment complets).

Lorsque cette thèse a débuté, on pouvait s'interroger, à la lumière des travaux existants dans ce domaine [AR00, MMS03, BPW03a], sur la possibilité d'obtenir des résultats de sûreté calculatoire pour des modèles logiques plus expressifs à base de théorie équationnelle, par exemple [AF01].

Dans le chapitre 5, nous avons entrepris d'étudier cette question d'un point de vue général pour le problème de l'équivalence statique. À l'aide du formalisme développé, nous avons présenté dans le chapitre 6 deux théories équationnelles pour lesquelles il est possible de prouver la sûreté calculatoire de l'équivalence statique, moyennant des hypothèses réalistes sur l'implémentation et les données considérées : d'une part la théorie du OU exclusif (pur), d'autre part une théorie regroupant trois formes de chiffrement ainsi que des clés faibles (typiquement des mots de passe).

Dans le second cas, le résultat obtenu constitue une justification partielle (*i.e.* dans le cas passif) de la définition des attaques par dictionnaire utilisée dans la première partie de cette thèse (chapitre 3), ou encore dans la procédure de B. Blanchet,

M. Abadi et C. Fournet [BAF05]. Il s'agit ainsi de la première définition logique des attaques par dictionnaire à bénéficier de procédures d'analyse dans le cas actif et d'une justification calculatoire partielle.

En guise de travaux futurs, il serait intéressant d'étudier d'autres théories équationnelles comme la réunion des théories considérées (OU exclusif + chiffrements) ou encore une théorie de l'exponentiation modulaire de type Diffie-Hellman.

De manière plus ambitieuse, on peut également s'interroger sur l'existence d'un modèle équationnel calculatoirement sûr pour un attaquant actif. À cet égard, les résultats négatifs de M. Backes et B. Pfitzmann [BP05], concernant le OU exclusif dans le formalisme (plus exigeant) de la simulabilité, sont naturellement un élément à prendre en considération.

Bibliographie

[AB05a] Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. *Journal of the ACM*, 52(1) :102–146, 2005.

[AB05b] Xavier Allamigeon and Bruno Blanchet. Reconstruction of attacks against cryptographic protocols. In *Proc. 18th IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 140–154, 2005.

[Aba99] Martín Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5), 1999.

[ABS05] Pedro Adão, Gergei Bana, and Andre Scedrov. Computational and information-theoretic soundness and completeness of formal encryption. In *Proc. 18th IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 170–184, 2005.

[ABW06] Martín Abadi, Mathieu Baudet, and Bogdan Warinschi. Guessing attacks and the computational soundness of static equivalence. In *Proc. 9th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'06)*, volume 3921 of *Lecture Notes in Computer Science*, pages 398–412, March 2006.

[AC04] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. In *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *Lecture Notes in Computer Science*, pages 46–58, 2004.

[AC05] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under (many more) equational theories. In *Proc. 18th IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 62–76, 2005.

[AC06] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 367(1–2) :2–32, 2006.

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communications. In *Proc. 28th Annual ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, 2001.

[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols : the Spi calculus. In *Proc. 4th ACM Conference on Computer and Communications Security (CCS'97)*, pages 36–47, 1997.

[AG98] Martín Abadi and Andrew D. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4) :267–303, 1998.

[AHBS05] Pedro Adão, Jonathan Herzog, Gergei Bana, and Andre Scedrov. Soundness of formal encryption in the presence of key-cycles. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, volume 3679 of *Lecture Notes in Computer Science*, pages 374–396, 2005.

[AL00] Roberto M. Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. In *Proc. 11th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *Lecture Notes in Computer Science*, pages 380–394, 2000.

[AN96] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions Software Engineering*, 22(1) :6–15, 1996.

[AR00] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (The computational soundness of formal encryption). In *Proc. 1st IFIP International Conference on Theoretical Computer Science (IFIP–TCS'00)*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22, 2000.

[AR02] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (The computational soundness of formal encryption). *Journal of Cryptology*, 15(2) :103–127, 2002.

[AW05] Martín Abadi and Bogdan Warinschi. Password-based encryption analyzed. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 664–676, 2005.

[BAF05] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. In *Proc. 20th IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 331–340, 2005.

[BAN90] Michael Burrows, Martín Abadi, and Roger M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1) :18–36, 1990.

[Bau] Mathieu Baudet. Prototype YAPA (Yet Another Protocol Analyzer). `http://www.lsv.ens-cachan.fr/~baudet/yapa/`.

[Bau05] Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 16–25, Alexandria, Virginia, USA, November 2005. ACM Press.

[BB05] Michele Boreale and Maria Grazia Buscemi. A method for symbolic analysis of security protocols. *Theoretical Computer Science*, 338(1–3) :393–425, 2005.

[BCK05] Mathieu Baudet, Véronique Cortier, and Steve Kremer. Computationally sound implementations of equational theories against passive adversaries. In *Proc. 32nd International Colloquium on Automata, Lan-*

guages and Programming (ICALP'05), volume 3580 of *Lecture Notes in Computer Science*, pages 652–663, 2005.

[BDNN02] Chiara Bodei, Pierpaolo Degano, Flemming Nielson, and Hanne Riis Nielson. Flow logic for Dolev-Yao secrecy in cryptographic processes. *Future Generation of Computer Systems*, 18(6) :747–756, 2002.

[BdNP99] Michele Boreale, Rocco de Nicola, and Rosario Pugliese. Proof techniques for cryptographic processes. In *Proc. 14th IEEE Symposium on Logics in Computer Science (LICS'99)*, pages 157–166, 1999.

[BL06] Michael Backes and Peeter Laud. Computationnally sound secrecy proofs by mechanized flow analysis. In *Proc. 13th ACM Conference on Computer and Communications Security (CCS'06)*, 2006. To appear.

[Blaa] Bruno Blanchet. Cryptoverif tool. `http://www.di.ens.fr/~blanchet/cryptoc.html`.

[Blab] Bruno Blanchet. ProVerif tool. `http://www.di.ens.fr/~blanchet/crypto.html`.

[Bla01] Bruno Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 82–96, 2001.

[Bla02] Bruno Blanchet. From secrecy to authenticity in security protocols. In *Proc. 9th International Static Analysis Symposium (SAS'02)*, volume 2477 of *Lecture Notes in Computer Science*, pages 342–359, 2002.

[Bla04] Bruno Blanchet. Automatic proof of strong secrecy for security protocols. In *Proc. 25th IEEE Symposium on Security and Privacy (SSP'04)*, pages 86–100, 2004.

[Bla06] Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *IEEE Symposium on Security and Privacy*, Oakland, California, May 2006. IEEE Computer Society Press. To appear.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange : Password-based protocols secure against dictionary attacks. In *Proc. 1992 IEEE Symposium on Security and Privacy (SSP'92)*, pages 72–84, 1992.

[BN05] Johanned Borgström and Uwe Nestmann. On bisimulations for the spi-calculus. *Mathematical Structures in Computer Science*, 15(3) :487–552, 2005.

[Bol97] Dominique Bolignano. Towards the formal verification of electronic commerce protocols. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW'97)*, pages 113–147, 1997.

[Bor01] Michele Boreale. Symbolic trace analysis of cryptographic protocols. In *Proc. 28th International Conference Automata Languages and Programming (ICALP'01)*, volume 2076 of *Lecture Notes in Computer Science*, pages 667–681, 2001.

[BP04] Michael Backes and Birgit Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. 17th IEEE Computer Science Foundations Workshop (CSFW'04)*, pages 204–218, 2004.

[BP05] Michael Backes and Birgit Pfitzmann. Limits of the cryptographic realization of Dolev-Yao style XOR. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, volume 3679 of *Lecture Notes in Computer Science*, pages 336–354, 2005.

[BP06] Bruno Blanchet and David Pointcheval. Automated security proofs with sequences of games. In *Proc. 26th Annual International Cryptology Conference (CRYPTO'06)*, Lecture Notes in Computer Science, 2006. To appear.

[BPW03a] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A composable cryptographic library with nested operations. In *Proc. 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 220–230, 2003.

[BPW03b] Michael Backes, Birgit Pfitzmann, and Michael Waidner. Symmetric authentication within simulatable cryptographic library. In *Proc. 8th European Symposium on Research in Computer Security (ESORICS'03)*, Lecture Notes in Computer Science, pages 271–290, 2003.

[BR03] Mihir Bellare and Phil Rogaway. Introduction to modern cryptography, 2003. `http://www.cs.ucsd.edu/~mihir/cse207/classnotes.html`.

[CB83] Jaques Corbin and Michel Bidoit. A rehabilitation of robinson's unification algorithm. In *IFIP Congress*, pages 909–914, 1983.

[CCM01] Hubert Comon, Véronique Cortier, and John C. Mitchell. Tree automata with one memory, set constraints and ping-pong protocols. In *Proc. 28th International Conference Automata Languages and Programming (ICALP'01)*, volume 2076 of *Lecture Notes in Computer Science*, pages 682–693, 2001.

[CDE04] Ricardo Corin, Jeroen Doumen, and Sandro Etalle. Analysing password protocol security against off-line dictionary attacks. In *Proc. 2nd International Workshop on Security Issues with Petri Nets and other Computational Models (WISP'04)*, volume 121 of *Electronic Notes in Theoretical Computer Science*, pages 47–63, 2004.

[CDL06] Véronique Cortier, Stéphanie Delaune, and Pascal Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1) :1–43, 2006.

[CH06] Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of mutual authentication and key-exchange protocols (extended abstract). In *Proc. 3rd Theory of Cryptography Conference (TCC'06)*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403, 2006.

[CJ97] John Clark and Jeremy Jacob. A survey of authentication protocol literature, 1997.

[CJ03] Hubert Comon and Jean-Pierre Jouannaud. Les termes en logique et en programmation (version préliminaire). Technical report, CNRS et Université de Paris Sud, 2003. `http://www.lri.fr/~jouannau/biblio.html`.

[CKRT03a] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST–TCS'03)*, volume 2914 of *Lecture Notes in Computer Science*, pages 124–135, 2003.

[CKRT03b] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. An NP decision procedure for protocol insecurity with XOR. In *Proc. 18th IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 261–270, 2003.

[CLS03] Hubert Comon-Lundh and Vitaly Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. 18th IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 261–270, 2003.

[CMAFE03] Ricardo Corin, Sreekanth Malladi, Jim Alves-Foss, and Sandro Etalle. Guess what ? Here is a new tool that finds some new guessing attacks. In *Proc. Workshop on Issues in the Theory of Security (WITS'03)*, pages 62–71, 2003.

[CMMU00] Evelyne Contejean, Claude Marché, Benjamin Monate, and Xavier Urbain. *The CiME Rewrite Tool*, 2000. `http://cime.lri.fr`.

[CMR01] Véronique Cortier, Jonathan K. Millen, and Harald Rueß. Proving secrecy is easy enough. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 97–110, 2001.

[Coh02] Ernie Cohen. Proving protocols safe from guessing. In *Proc. Foundations of Computer Security (FCS'02)*, pages 85–92, 2002.

[Cor02] Véronique Cortier. Observational equivalence and trace equivalence in an extension of the spi-calculus. Technical report, Laboratoire Spécification et Vérification, 2002.

[CR05] Yannick Chevalier and Michaël Rusinowitch. Combining intruder theories. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 639–651, 2005.

[CW05] Véronique Cortier and Bogdan Warinschi. Computationally sound, automated proofs for security protocols. In *European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171, Edinburgh, UK, 2005.

[DDM+05] Anupam Datta, Ante Derek, John C. Mitchell, Vitaly Shmatikov, and Mathieu Turuani. Probabilistic Polynomial-time Semantics for a Protocol Security Logic. In *Proc. of 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2005. Lisboa, Portugal.

[Del06a] Stéphanie Delaune. An undecidability result for AGh. Research report, Laboratoire Spécification et Vérification, ENS Cachan, 2006.

[Del06b] Stéphanie Delaune. *Vérification des protocoles cryptographiques et propriétés algébriques*. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, 2006.

[Der87] Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3 :69–116, 1987.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Society*, 22(6) :644–654, 1976.

[DJ04a] Stéphanie Delaune and Florent Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In *Proc. 11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 278–287, 2004.

[DJ04b] Stéphanie Delaune and Florent Jacquemard. A theory of dictionary attacks and its complexity. In *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, pages 2–15, 2004.

[DLLT06] Stéphanie Delaune, Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. Symbolic protocol analysis in presence of a homomorphism operator and exclusive or. In *Proc. 33rd International Colloquium on Automata, Languages and Programming (ICALP'06)*, volume 4052 of *Lecture Notes in Computer Science*, pages 132–141, Venice, Italy, 2006.

[DMP03] Nancy A. Durgin, John Mitchell, and Dusko Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11(4) :677–722, 2003.

[Dow01] Gilles Dowek. Higher-order unification and matching. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 1009–1062. Elsevier and MIT Press, 2001.

[DS81] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8) :533–536, 1981.

[DSV03] Luca Durante, Riccardo Sisto, and Adriano Valenzano. Automatic testing equivalence verification of spi calculus specifications. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12(2) :222–284, 2003.

[DV96] Anatoli Degtyarev and Andrei Voronkov. The undecidability of simultaneous rigid e-unification. *Theoretical Computer Science*, 166(1–2) :291–300, 1996.

[DvOW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2) :107–125, 1992.

[DY83] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12) :198–208, 1983.

[EG83] Shimon Even and Oded Goldreich. On the security of multi-party ping-pong protocols. In *Proc. 24th IEEE Annual Symposium on Foundations of Computer Science (FOCS'83)*, pages 34–39, 1983.

[Fou02] Cédric Fournet. Private communication, 2002.

[GLNS93] Li Gong, Mark A. Lomas, Roger M. Needham, and Jerome H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5) :648–656, 1993.

[Gou00] Jean Goubault-Larrecq. A method for automatic cryptographic protocol verification (extended abstract). In *Proc. of the Workshops of the 15th International Parallel and Distributed Processing Symposium*, volume 1800 of *Lecture Notes in Computer Science*, pages 977–984, 2000.

[Gou04] Jean Goubault-Larrecq. Une fois qu'on n'a pas trouvé de preuve, comment le faire comprendre à un assistant de preuve ? In *Actes des 15èmes Journées Francophones sur les Langages Applicatifs (JFLA'04)*, pages 1–40. INRIA, 2004. Invited paper.

[GRV05] Jean Goubault-Larrecq, Muriel Roger, and Kumar N. Verma. Abstraction and resolution modulo AC : How to verify Diffie-Hellman-like protocols automatically. *Journal of Logic and Algebraic Programming*, 64(2) :219–251, 2005.

[Hoh03] Susan Hohenberger. The cryptographic impact of groups with infeasible inversion. Master's thesis, MIT, 2003.

[Hue75] Gérard Huet. A unification algorithm for typed λ-calculus. *Theoritical Computer Science*, 1 :27–57, 1975.

[Hui99a] Antti Huima. Efficient infinite-state analysis of security protocols. In *Proc. FLOC Workshop on Formal Methods and Security Protocols*, 1999.

[Hui99b] Antti Huima. Efficient infinite-state analysis of security protocols. In *Proc. FLOC Workshop on Formal Methods and Security Protocols(FMSP'99)*, 1999.

[Hul80] Jean-Marie Hullot. Canonical forms and unification. In *Proc. 8th International Conference on Automated Deduction (CADE'80)*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334, 1980.

[Hüt00] Hanz Hüttel. Deciding framed bisimilarity. In *Proc. 4th Int. Workshop on Verification of Infinite State Systems(INFINITY'02)*, pages 1–20, 2000.

[JK91] Jean-Pierre Jouannaud and Claude Kirchner. Solving equations in abstract algebras : A rule-based survey of unification. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic : Essays in Honor of Alan Robinson*, pages 257–321. MIT Press, 1991.

[JLM05] Romain Janvier, Yassine Lakhnech, and Laurent Mazaré. Completing the picture : Soundness of formal encryption in the presence of active adversaries. In *European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 172–185, 2005.

[KK05] Detlef Käeler and Ralf Küsters. Constraint solving for contract-signing protocols. In *Proc. 16th International Conference on Concurrency Theory*, volume 3653 of *Lecture Notes in Computer Science*, pages 233–247, 2005.

[KR02] Steve Kremer and Jean-François Raskin. Game analysis of abuse-free contract signing. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 206–222, 2002.

[KR05] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, 2005.

[Kre06] Steve Kremer. Private communication, November 2006.

[Lau02] Peeter Laud. *Computationally Secure Information Flow.* PhD thesis, Universität des Saarlandes, 2002.

[Lau04] Peeter Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proc. IEEE Symposium on Security and Privacy (SSP'04)*, pages 71–85, 2004.

[LC03] Peeter Laud and Ricardo Corin. Sound computational interpretation of formal encryption with composed keys. In *Proc. 6th International Conference on Information Security and Cryptology (ICISC'03).* KIISC, 2003.

[LMMS98] Patrick Lincoln, John C. Mitchell, Mark Mitchell, and Andre Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proc. 5th ACM Conference on Computer and Communications Security (CCS'98)*, pages 112–121, 1998.

[Low96] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *Proc. 2nd International Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS'96)*, pages 147–166, 1996.

[Low97] Gavin Lowe. A hierarchy of authentication specifications. In *Proc. 10th Computer Security Foundations Workshop (CSFW'97)*, pages 31–44, 1997.

[Low04] Gavin Lowe. Analysing protocols subject to guessing attacks. *Journal of Computer Security*, 12(1) :83–98, 2004.

[Mea96] Catherine Meadows. The NRL protocol analyzer : An overview. *Journal of Logic Programming*, 26(2) :113–131, 1996.

[Mea99] Catherine Meadows. Analysis of the Internet Key Exchange protocol using the NRL protocol analyzer. In *Proc. IEEE Symposium on Security and Privacy (SSP'99)*, pages 216–231, 1999.

[MH94] Aart Middeldorp and Erik Hamoen. Completness results for basic narrowing. *Applicable Algebra in Engineering, Communication and Computing*, 5 :213–253, 1994.

[Mic05] Daniele Micciancio. The rsa group is pseudo-free. In *Advances in Cryptology – EUROCRYPT'05*, volume 3494 of *Lecture Notes in Computer Science*, pages 387–403, 2005.

[Mil93] Robin Milner. The polyadic pi-calculus : a tutorial. *Logic and Algebra of Specification*, pages 203–246, 1993.

[MMS97] John C. Mitchell, Mark Mitchell, and Ulrich Stern. Automated analysis of cryptographic protocols using Murϕ. In *Proc. IEEE Symposium on Security and Privacy (SSP'97)*, pages 141–153, 1997.

[MMS03] Paulo Mateus, John C. Mitchell, and Andre Scedrov. Composition of cryptographic protocols in a probabilistic polynomial-time process calculus. In *Proc. 14th International Conference on Concurrency Theory*

(CONCUR'03), volume 2761 of *Lecture Notes in Computer Science*, pages 323–345, 2003.

[Mon03] David Monniaux. Abstracting cryptographic protocols with tree automata. *Science of Computer Programming*, 47(2–3) :177–202, 2003.

[MRST01] John C. Mitchell, Ajith Ramanathan, Andre Scedrov, and Vanessa Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols. In *Proc. 17th Annual Conference on the Mathematical Foundations of Programming Semantics*, volume 45 of *Electronic Notes in Theoretical Computer Science*, Aarhus, Denmark, 2001.

[MS01] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, pages 166–175, 2001.

[MS03] Jonathan K. Millen and Vitaly Shmatikov. Symbolic protocol analysis with products and Diffie-Hellman exponentiation. In *Proc. 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, pages 47–61, 2003.

[MvV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[MW04a] Daniele Micciancio and Bogdan Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1) :99–129, 2004.

[MW04b] Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proc. 1st Theory of Cryptography Conference (TCC'04)*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151, 2004.

[NS78] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12) :993–999, 1978.

[OND98] Friedrich Otto, Paliath Narendran, and Daniel J. Dougherty. Equational unification, word unification, and 2nd-order equational unification. *Theoretical Computer Science*, 198(1-2) :1–47, 98.

[Pau98] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6 :85–128, 1998.

[Pot02] François Pottier. A simple view of type-secure information flow in the π-calculus. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 320–330, June 2002.

[PP04] Duong Hieu Phan and David Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In *Proc. Selected Areas in Cryptography (SAC'04)*, volume 3357 of *Lecture Notes in Computer Science*, pages 185–200, 2004.

[PS03] François Pottier and Vincent Simonet. Information flow inference for ML. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 25(1) :117–158, January 2003.

[Riv04] Ronald L. Rivest. On the notion of pseudo-free groups. In *Proc. 1st Theory of Cryptography Conference (TCC'04)*, volume 2951 of *Lecture Notes in Computer Science*, pages 505–521, 2004.

[RMST04] Ajith Ramanathan, John C. Mitchell, Andre Scedrov, and Vanessa Teague. Probabilistic bisimulation and equivalence for security analysis of network protocols. In *Proc. 7th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'04)*, volume 2987 of *Lecture Notes in Computer Science*, pages 468–483, 2004.

[Sho04] Victor Shoup. Sequences of games : a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004.

[SRA81] Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman. *The Mathematical Gardner*, chapter Mental Poker, pages 37–43. Prindle, Weber, and Schmidt, 1981.

[THG99] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces : Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.

[War03] Bogdan Warinschi. A computational analysis of the Needham-Schroeder protocol. In *Proc. 16th IEEE Computer Science Foundations Workshop (CSFW'03)*, pages 248–262, 2003.

[Wik] Wikipedia, the Free Encyclopedia. `http://www.wikipedia.org`.

[WL93] Thomas Y. C. Woo and Simon S. Lam. A semantic model for authentication protocols. In *Proc. IEEE Symposium on Research in Security and Privacy (SP'93)*, pages 178–195, 1993.

[ZD06] Roberto Zunino and Pierpaolo Degano. Handling exp, $\times$ (and timestamps) in protocol analysis. In *Proc. 9th Int. Conference on Foundations of Software Science and Computation Structures (FOSSACS'06)*, volume 3921 of *Lecture Notes in Computer Science*, pages 413–427, 2006.

Table des figures

Liste des tableaux