Algorithme de décision de l'équivalence symbolique de systèmes de contraintes

V. Cheval, sous la direction de H. Comon-Lundh, S. Delaune, SECSI / LSV

23 août 2009

Le contexte général

La sécurisation des communications qui passent par des canaux publics, comme internet ou les réseaux sans fil, est habituellement assurée par des protocoles cryptographiques. Leur utilisation massive accroit le besoin de s'assurer que ces protocoles satisfont bien les propriétés de sécurité attendues. Nous nous intéressons ici uniquement aux failles conceptuelles, qui permettent des attaques, même en présence d'une cryptographie parfaite [1].

La plupart des travaux de recherche [12, 3, 14, 16, 5, 7, 11, 9] se sont focalisés sur les propriétés de trace.Or il existe plusieurs propriétés de sécurité qui ne sont pas des propriétés de trace, mais qui s'expriment à l'aide de l'équivalence observationnelle. Un exemple typique est la propriété d'anonymat dans les protocoles de vote. Informellement, l'équivalence observationnelle de deux processus décrit l'impossibilité, pour un attaquant actif, de distinguer ces deux processus quelles que soient les actions qu'il effectue.

Le problème étudié

Notre objectif est de mettre au point des techniques d'analyse automatique de protocoles qui s'appliquent aux propriétés d'équivalence. Il existe trois travaux principaux qui tentent de répondre à cette question.

Tout d'abord, [13] donne le premier un algorithme de décision de l'équivalence observationnelle pour des processus bornés. Mais la procédure présente plusieurs inconvénients : elle est spécialisée pour certaines primitives cryptographiques et elle est d'une compléxité très élevée. Ensuite, [6] propose une notion approchée de l'équivalence observationnelle qui peut être codée et vérifiée automatiquement par le logiciel ProVerif [5]. Néanmoins, il n'y a aucune garantie de terminaison ni de correction (possibilité de fausses attaques). Enfin, M. Baudet [4] donne une procédure correcte, complète (et qui

termine) pour l'équivalence symbolique de processus bornés dans le pi-calcul appliqué. Complété par celui de V. Cortier et S. Delaune [10], ce travail permet de décider l'équivalence observationnelle d'une classe de processus positifs.

La contribution proposée

Notre travail consiste à reprendre le travail de [4], mais dans le formalisme de [9] : les contraintes de déductibilité . Nous proposons une autre procédure, qui est moins compliquée que [4], extensible (par exemple aux processus non positifs) et implémentable.

Notre algorithme va résoudre *simultanément* les contraintes de déductibilité correspondant à l'exécution des deux processus. Ceci nous permet de réduire l'équivalence de contraintes symboliques à l'équivalence des *formes résolues* de [9].

Par contre, pour des raisons de simplicité, nous n'avons considéré ici qu'un ensemble fixe de primitives : chiffrement symétrique et appariement.

Les arguments en faveur de sa validité

Nous proposons un schéma général de règles de simplification qui préserve l'équivalence symbolique des systèmes de contraintes. Un des points forts de ce rapport est que notre ensemble de règles est aisément modifiable : pour étendre la théorie de l'attaquant, il suffit de montrer que les nouvelles règles rentrent de le shéma général. Un autre point fort est que nos résultats sont indépendants de la stratégie d'utilisation des règles. Il est donc possible d'optimiser l'algorithme selon le type de problème et les ressources, tout en conservant l'équivalence symbolique.

Enfin, nous avons prouvé correction et complétude des règles et une implémentation ainsi que la preuve de la terminaison sont en cours.

Le bilan et les perspectives

Le travail effectué durant ce stage n'est qu'une étape. La prochaine étape consistera à résoudre le problème de l'équivalence symbolique sur ces systèmes de contraintes simplifiés. Nous devrons ensuite généraliser à d'autres primitives cryptographiques et à d'autres classes de processus. Enfin, il nous faut compléter une réalisation logicielle qui déterminera la pertinence de l'approche en pratique et finir la preuve de terminaison.

1 Introduction

1.1 Contexte général

De plus en plus de données confidentielles transitent par des canaux qui ne sont pas sûrs comme internet, ou des réseaux sans fil. Pour sécuriser ces communications, on emploie aujourd'hui massivement les *protocoles cryptographiques*. Ce sont de petits programmes distribués, qui utilisent des primitives cryptographiques.

Pour augmenter notre confiance, par exemple dans le cas du vote électronique dont les enjeux sont cruciaux, il est important de pouvoir *prouver* la correction de ces protocoles.

Nous ne considérons pas ici les attaques sur les primitives cryptographiques : nous supposons que la cryptographie est parfaite. Autrement dit, nous ne considérons qu'un modèle symbolique "idéal". Il existe cependant de nombreux travaux (par exemple [8]) qui établissent les conditions dans lesquelles ce modèle idéal est fidèle.

De la même façon, nous ne considérons pas les attaques qui se fondent sur des erreurs de programmation : nous ne considérons que la spécification du protocole et supposons que sa réalisation logicielle satisfait les contraintes de la spécification

Même en supposant la cryptographie parfaite et le programme sans erreur, il existe de nombreuses attaques [1], et il est donc important de s'intéresser à la sécurité à ce niveau d'abstraction. C'est ce que nous faisons dans ce mémoire.

1.2 Le problème étudié

La plupart des travaux de recherche [12, 3, 14, 16, 5, 7, 11, 9] se sont focalisés sur les propriétés de trace, comme par exemple :

- le *secret simple*, qui énonce qu'un message donné ne peut être récupéré en totalité par l'attaquant.
- les propriétés d'*authenticité*, qui expriment un accord sur un message entre deux partenaires distincts.

Or il existe plusieurs propriétés de sécurité qui ne sont pas des propriétés de trace, mais qui s'expriment à l'aide de l'équivalence observationnelle. Un exemple typique est la propriété d'anonymat dans les protocoles de vote. L'anonymat requiert que la liaison entre a et $v(a)$ ne soit pas possible, ce qu'on peut exprimer à l'aide de l'équivalence observationnelle $\sim_o$ par $P(a, v(a), b, v(b)) \sim_o P(a, v(b), b, v(a))$: un adversaire ne peut pas distinguer entre une instance du protocole de vote P dans laquelle a vote $v(a)$ et

b vote $v(b)$, d'une instance du protocole dans laquelle a vote $v(b)$ et b vote $v(a)$.

Notre objectif est de mettre au point des techniques d'analyse automatique de protocoles, qui ne s'appliquent pas seulement aux propriétés de trace, mais aussi aux propriétés d'équivalence. Il existe trois travaux principaux qui tentent de répondre à cette question.

- Tout d'abord, [13] donne le premier un algorithme de décision de l'équivalence observationnelle pour des processus bornés. Mais la procédure présente plusieurs inconvénients :
  1. elle est limitée au spi-calcul, donc à des primitives fixées
  2. elle n'est pas implémentable (d'une complexité multi-exponentielle)
  3. elle ne permet pas de traiter les propriétés de trace.
- B. Blanchet, M. Abadi et C. Fournet dans [6] proposent une notion approchée de l'équivalence observationnelle (la "diff-équivalence") qui peut être codée et vérifiée automatiquement par le logiciel ProVerif [5]. Le point fort de cette approche est qu'elle ne suppose pas les processus bornés (il peut y avoir des réplications). Par contre, il s'agit seulement d'une équivalence approchée et de plus il n'y a aucune garantie de terminaison ni de correction (possibilité de fausses attaques).
- M. Baudet [4] donne une procédure correcte, complète (et qui termine) pour l'équivalence symbolique de processus bornés dans le pi-calcul appliqué. Complété par les travaux de V. Cortier et S. Delaune [10], ce résultat permet de décider l'équivalence observationnelle d'une classe de processus positifs (pas de branche else ni de test négatif).

Notre travail consiste à reprendre le travail de [4], mais dans un autre formalisme, celui des contraintes de déductibilité [15]. Il s'agit donc de proposer une autre procédure, qui est moins compliquée que [4], extensible (par exemple aux processus non positifs) et implémentable.

1.3 Notre contribution

Tout d'abord, nous appuyant sur [10], nous nous intéressons à l'équivalence de traces. Même pour des processus bornés (c'est à dire dont le nombre d'étapes de calcul est borné), l'ensemble des traces est infini à cause des choix non bornés de l'attaquant. Nous avons choisi de le représenter à l'aide de systèmes de contraintes de déductibilité : nous adoptons le formalisme proposé par H. Comon-Lundh, V. Cortier et E. Zalinescu dans [9]. L'idée principale de ce formalisme est de remplacer les messages reçus par des variables, en

contraignant celles-ci à être déductibles par l'intrus : l'attaquant peut intercepter des messages et en forger de nouveaux qui sont inconnus et donc représentés par des variables, mais il doit être capable de les fabriquer à partir des messages qui lui sont connus. Les participants honnêtes répondent alors, seulement si les tests qu'ils effectuent sur ces variables sont satisfaits. La réponse, qui dépend des variables précédentes, va enrichir la connaissance de l'intrus. Une *solution* est une affectation de messages aux variables, qui satisfait toutes les contraintes et qui viole la propriété de sécurité. Une solution correspond alors à une attaque : c'est un choix de messages forgés par l'attaquant, qui permet de réaliser une séquence d'actions conduisant à une violation de la sécurité.

Cette notion de solution n'est pas suffisante lorsqu'on s'intéresse à l'équivalence observationnelle et plus seulement aux propriétés de trace. Prenons deux exemples de séquences de trois messages $a, enc(b, a), b$ d'une part et a, b, b d'autre part (en supposant un chiffrement symétrique). Les messages déductibles sont identiques dans les deux exemples. Pourtant les deux séquences sont distingables par un attaquant. Celui ci peut en effet observer que les deux derniers messages sont distincts dans la première séquences, alors qu'ils sont identiques dans la deuxième. Nous avons donc besoin d'une notion plus fine de solutions de systèmes de contraintes, dans laquelle on tient également compte de la *manière* dont les calculs sont effectués par l'attaquant ; pour obtenir l'équivalence il faut que les mêmes actions de l'attaquant conduisent à des résultats indistingables.

Plus précisément, nous commençons par utiliser le résultat de [10], pour nous ramener à l'équivalence de traces : nous ne considérons de ce fait qu'une sous-classe de processus bornés. De plus, nous ne considérons qu'un ensemble très simple de primitives cryptographiques : le chiffrement symétrique et l'appariement (mais nous ne supposons pas que les clefs sont atomiques ; celles-ci peuvent elles-mêmes être des chiffrements). Ces restrictions ne sont pas nécessairement essentielles. Nous les avons prises pour nous concentrer sur les problèmes principaux.

L'équivalence de deux processus se réduit alors (comme dans [4]) à l'égalité des solutions des contraintes de déductibilité associées aux deux processus. Mais la notion de solution tient compte désormais des recettes.

Notre principale contribution est alors de proposer un algorithme qui va résoudre *simultanément* les contraintes de déductibilité correspondant à l'exécution des deux processus. Ceci nous permet de réduire l'équivalence de contraintes symboliques à l'équivalence des *formes résolues* de [9]. Les règles de [9] sont insuffisantes pour deux raisons : d'abord elles ne préservent pas nécéssairement la notion plus générale de solution dont nous avons besoin.

Ensuite, et c'est là la difficulté principale, il se peut que les règles s'appliquent sur un des systèmes et pas l'autre. Nous en verrons des exemples, mais c'est typiquement le cas quand, à des positions qui se correspondent, on trouve une variable dans l'un des deux systèmes et un terme non variable dans l'autre système. Ces situations peuvent forcer à expanser les variables, ce qui pose alors un problème de terminaison.

> *Nous proposons un schéma général de règles de simplification qui préserve l'équivalence symbolique des systèmes de contraintes qui se termine toujours et qui permet de se ramener à l'équivalence de formes résolues.*

Un des points forts de ce rapport est que notre ensemble de règles est aisément modifiable : pour étendre la théorie de l'attaquant, il suffit de montrer que les nouvelles règles rentrent de le shéma général.

Un autre point fort est que nos résultats sont indépendants de la stratégie d'utilisation des règles. Il est donc possible d'optimiser l'algorithme selon le type de problème et les ressources, tout en conservant l'équivalence symbolique.

1.4 Perspectives

Une implémentation de l'algorithme est en cours de réalisation. Il reste à la terminer, à finir la preuve de terminaison et aussi à donner un algorithme pour décider l'équivalence des formes résolues. Parmi les extensions envisagées, citons :

- extension aux processus non-positifs
- extension à d'autres primitives cryptographiques
- algorithme de décision de la bissimimulation étiquetée (évitant de passer par l'équivalence de trace)

1.5 Plan du mémoire

Nous ne donnons pas ici la définition de la classe de processus considérée, ni la justification de la traduction en systèmes de contraintes. Pour cela, nous nous référons à [10, 4]. Nous nous concentrons sur le problème de l'équivalence symbolique et de sa solution.

Nous commençons par énoncer précisément le problème dans les paragraphes 2 et 3, puis nous donnons nos règles de simplification de contraintes dans le paragraphe 4 et énonçons les principaux résultats. Les preuves sont données en annexe.

2 Syntaxe et sémantique

2.1 Termes et Recettes

Dans ce rapport, nous considérons un ensemble fixe de primitives : le système dit de "Dolev-Yao" pour le chiffrement symétrique. Dans ce modèle, les messages sont des termes construits sur les symboles de fonctions suivants (dont l'arité est indiquée entre parenthèse) :

- les constructeurs $\mathcal{C} = \{\langle\rangle(2); enc(2)\}$. $\langle m_1, m_2\rangle$ représente la paire des deux messages m_1 et m_2. $enc(m_1, m_2)$ représente le chiffrement de m_1 par m_2.
- les destructeurs $\mathcal{D} = \{\pi_1(1); \pi_2(1); dec(2)\}$. π_1, π_2 sont les deux projections qui permettent de récupérer les composantes d'une paire. $dec(m_1, m_2)$ représente le déchiffrement de m_1 par m_2.
- les constantes privées $\mathcal{N}$ (ensemble non borné)
- les variables standard $\mathcal{X}_v$ (ensemble non borné)

$\mathcal{T}(\mathcal{F}, X)$ est l'ensemble des termes construits sur les symboles de fonction $\mathcal{F}$ et les variables X.

Les destructeurs sont définis via le système de réécriture :

$$\pi_1(\langle x, y\rangle) \to x \qquad \pi_y(\langle x, y\rangle) \to y \qquad dec(enc(x, y), y) \to x.$$

Ce système de réécriture est convergent : on note $t\downarrow$ l'unique forme normale d'un terme t.

Pour définir les recettes (actions de l'attaquant), nous utilisons un ensembe de variables spéciales $\mathcal{AX} = \{ax_i \mid i \in \mathbb{N}\}$. Elles sont utilisées uniquement comme des pointeurs vers un des messages de la liste à disposition de l'attaquant. Un exemple typique de recette est : “prendre le premier message et le déchiffrer avec le troisième” ce qui s'écrira $dec(ax_1, ax_3)$.

Définition 2.1 (Recette, Preuve)**.** L'ensemble des *recettes* (ou *preuves*), noté Π, est l'ensemble $\mathcal{T}(\mathcal{C} \cup \mathcal{D}, \mathcal{AX})$.

Soit $S = [s_1, \ldots, s_n]$ une séquence de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$ et $\xi \in \Pi$, on définit $\xi[S]$ par récurrence sur ξ :

- $ax_i[S] = s_i$ si $i \leq n$
- $ax_i[S]$ n'est pas défini sur $i > n$
- $\forall f \in \mathcal{C} \cup \mathcal{D}, f(\xi_1, ..., \xi_m)[S] = f(\xi_1[S], ..., \xi_m[S])$

Un exemple est donné en annexe B.1.

Soit $\xi \in \Pi$, on note $St(\xi)$ l'ensemble des sous-preuves de ξ.

Définition 2.2 (Terme à la position)**.** Soit $S = [t_1, \ldots, t_n]$ une séquence de termes de $\mathcal{T}(\mathcal{F}, X)$. Pour tout $p \in [1..n]$, *le terme à la position* p *dans* S, noté $S_{|p}$, est défini par $S_{|p} = t_p$.

Dans la suite nous dirons aussi que T_1 est *incluse* dans T_2, ce que nous notons $T_1 \sqsubseteq T_2$ si T_1 est un préfixe de T_2.

2.2 Systèmes de contraintes

Nous suivons ici le formalisme de [9].

Définition 2.3 (Contrainte)**.** Une *contrainte* est une expression $T \Vdash u$ où $u \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$, T est une séquence de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$

Intuitivement, T est la connaissance de l'attaquant à un instant donné et u est le terme (motif) que l'attaquant doit former pour donner le change aux agents honnêtes.

Un exemple est donné en annexe B.2.

Définition 2.4 (Système de contraintes)**.** Un *système de contraintes* C est soit $\bot$, soit un couple $(\mathcal{C}_o, \mathcal{E}_q)$ vérifiant les propriétés suivantes :

- $\mathcal{C}_o$ est une séquence de contraintes $[T_1 \Vdash u_1; \ldots; T_n \Vdash u_n]$ avec $\forall i \in [1 \ldots n], u_i \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$ et T_i est une séquence de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$
- $\mathcal{E}_q$ est un ensemble d'équations $u = v$ et de disequations $u \neq v$ tq $\exists \sigma = mgu(\{u = v \mid (u = v) \in \mathcal{E}_q\})$ tq $Dom(\sigma) \cap Var(\mathcal{C}_o) = \varnothing \wedge CoDom(\sigma) \subseteq Var(\mathcal{C}_o) \wedge Var(\mathcal{E}_q) \subseteq Var(\sigma) \cup Var(\mathcal{C}_o)$.
- $T_1 \sqsubseteq \cdots \sqsubseteq T_n$
- $\forall i, \forall x \in Var(T_i), \exists j < i$ tq $x \in Var(u_j)$

On notera $T_{max}(C) \stackrel{def}{=} T_n$.

La condition $T_1 \sqsubseteq \cdots \sqsubseteq T_n$, appelée *monotonie* dans [9], exprime que la connaissance de l'intrus est croissante. La dernière condition exprime qu'une nouvelle variable est introduite pour chaque nouveau message émis. Remarquons que la dernière propriété force T_1 a être clos.

Exemple 2.1. Exemple de système de contraintes :

$$\begin{array}{lcl} a, b & \Vdash & x \\ a, b, \langle enc(x, c), b \rangle & \Vdash & enc(y, c) \end{array} \qquad \mathcal{E}_q = \{z = \langle x, a \rangle\}$$

Définition 2.5 (Solution d'un système de contraintes)**.** Soit $C = (\mathcal{C}_o, \mathcal{E}_q)$ un système de contraintes tel que $\mathcal{C}_o = [T_1 \Vdash u_1; \ldots; T_n \Vdash u_n]$. On dit que $(\sigma, \xi_1, ..., \xi_n)$ est une *solution* de C ssi :

– σ est une substitution de $\mathcal{X}_v$ dans $\mathcal{T}(\mathcal{C} \cup \mathcal{N})$
– $\forall i \in [1..n], \xi_i[T_i\sigma]\downarrow = u_i\sigma$
– $\forall (u = v) \in \mathcal{E}_q, u\sigma\downarrow = v\sigma\downarrow$.
– $\forall (u \neq v) \in \mathcal{E}_q, u\sigma\downarrow \neq v\sigma\downarrow$.

On note $Sol(C)$ l'ensemble des solutions du système de contraintes C. Si $(\sigma, \xi_1, ..., \xi_n)$ est solution de C, on dit que $(\xi_1, ..., \xi_n)$ est une *solution du second ordre* et σ est *une solution du premier ordre*. Si $C = \bot$, alors $Sol(C) = \varnothing$.

Exemple 2.2. On reprend l'exemple du système de contraintes défini dans l'exemple 2.1. Soit le triplet (σ, ξ_1, ξ_2) défini tel que :

$$\sigma = \{x \to \langle a, a\rangle; y \to \langle a, a\rangle; z \to \langle\langle a, a\rangle, a\rangle\}, \xi_1 = \langle ax_1, ax_1\rangle, \xi_2 = \pi_1(ax_3)$$

(σ, ξ_1, ξ_2) est solution du système de contraintes.

Pour pourvoir restreindre les solutions et assurer la terminaison, nous étendons la définition des systèmes de contraintes en ajoutant des annotations.

Définition 2.6 (Annotation)**.** Soit $C = ([T_1 \Vdash u_1; \ldots; T_n \Vdash u_n], \mathcal{E}_q)$ un système de contraintes. $\forall i \in [1..n], \forall p \in [1..|T_i|]$, soit $t = T_{i|p}$. Une *annotation* sur t est un élément $NoAx(u)$ où $u \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$.
$\forall i \in [1..n]$, une *annotation* sur u_i est l'un des éléments ci-dessous :
– $NoCons(u)$ avec $u \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$
– $NoPi(u)$ avec $u \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$
– $NoDec(u)$ avec $u \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$

On notera $Ann_{i,p}(C)$ l'ensemble des annotations du terme $T_{i|p}$. On notera $Ann_{i,0}(C)$ l'ensemble des annotations du terme u_i.

Informellement, si $t = T_{|p}$, l'annotation $NoAx(u)$ sur t empêche l'utilisation de l'axiome ax_p si t et u sont égaux après instanciation. L'annotation $NoPi(u)$ empêche l'utilisation des destructeurs π_1 et π_2 sur un terme égal à u après instanciation. L'annotation $NoDec(u)$ sur t empêche l'utilisation de *dec* sur un terme égal à u après instanciation. Et enfin, $NoCons(u)$ empêche l'utilisation d'un constructeur pour déduire un terme égal à u après instanciation.

Remarque 1. Soit t un terme et soit F l'ensemble des annotations de t, on notera t^F pour représenter t et F. On pourra aussi écrire simplement t pour le terme et son ensemble d'annotations.

Définition 2.7 (Système de contraintes annoté)**.** Un *système de contraintes annoté* $C = ([T_1 \Vdash u_1, \ldots, T_n \Vdash u_n], \mathcal{E}_q)$ est un système de contraintes dont les termes peuvent être annotés et où C vérifie les propriétés suivantes : $\forall u \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v), \forall i \in [1..n], \forall p \in [1..|T_i|]$,

- si $NoAx(u) \in Ann_{i,p}(C)$, alors $\forall j \in [1 \ldots n], NoAx(u) \in Ann_{j,p}(C)$ si $T_{j|_p}$ est bien définie.
- si $NoPi(u) \in Ann_i(C)$, alors $\forall j \in [1 \ldots n], NoPi(u) \in Ann_j(C)$.
- si $NoCons(u) \in Ann_i(C)$ alors $\forall j \in [1 \ldots n], NoCons(u) \in Ann_j(C)$.
- si $NoDec(u) \in Ann_i(C)$, alors $\forall j \leq i, NoDec(u) \in Ann_j(C)$.

Définition 2.8 (Solution d'un système de contraintes annoté)**.** Soit $C = (\mathcal{C}_o, \mathcal{E}_q)$ un système de contraintes annotés tel que $\mathcal{C}_o = [T_1 \Vdash u_1^{F_1}; \ldots; T_n \Vdash u_n^{F_n}]$. On dit que $(\sigma, \xi_1, ..., \xi_n)$ est une solution de C ssi :

- $(\sigma, \xi_1, ..., \xi_n)$ est une solution de C sans annotation.
- $\forall i \in [1..n], \forall p \in [1 \ldots |T_i|]$, si $T_{i|_p} = t^F$ et $NoAx(u) \in F$, alors $ax_p \in St(\xi_i) \Rightarrow t\sigma \neq u\sigma$
- $\forall i \in [1..n], NoDec(u) \in F_i \Rightarrow \forall dec(\xi, \xi') \in St(\xi_i), \xi[T_i\sigma]\downarrow \neq u\sigma$
- $\forall i \in [1..n], NoPi(u) \in F_i \Rightarrow \forall \pi_1(ax_p) \in St(\xi_i), ax_p[T_i\sigma]\downarrow \neq u\sigma \wedge \forall \pi_2(ax_p) \in St(\xi_i), ax_p[T_i\sigma]\downarrow \neq u\sigma$
- $\forall i \in [1..n], NoCons(u) \in F_i \Rightarrow \forall \xi \in St(\xi_i), Top(\xi) \in \mathcal{C} \Rightarrow \xi[T_i\sigma]\downarrow \neq u\sigma$

On dit alors que $(\sigma, \xi_1, ..., \xi_n)$ satisfait l'ensemble des annotations de C.

Un exemple est donné en annexe B.3.

Définition 2.9 (Forme résolue)**.** Soit $C = (\mathcal{C}_o, \mathcal{E}_q)$ un système de contraintes tel que $\mathcal{C}_o = [T_1 \Vdash u_1^{F_1}; \ldots; T_n \Vdash u_n^{F_n}]$. On dit que C est sous forme résolue ssi

- $\forall i \in [1..n], u_i \in \mathcal{X}_v$
- $\forall (i,j) \in [1..n]^2, i \neq j \Rightarrow u_i \neq u_j$

Un système de contraintes $C = \bot$ est toujours en forme résolue.

Définition 2.10 (Structure d'un système de contraintes)**.** Soient $C = ([T_1 \Vdash u_1; \ldots; T_n \Vdash u_n], \mathcal{E}_q)$ et $C' = ([T'_1 \Vdash u'_1; \ldots; T'_m \Vdash u'_m], \mathcal{E}'_q)$ deux systèmes de contraintes. C et C' ont la même *structure* ssi :

- $n = m$
- $\forall i \in [1..n], |T_i| = |T'_i|$

3 Equivalence

Dans toute la suite de ce rapport, tout les systèmes de contraintes sont des systèmes de contraintes annotés (sauf indication). De plus, quand on

parlera de couples $(S, C), (S', C')$ avec C, C' des systèmes de contraintes, et S, S' des séquences de termes, on supposera toujours que C et C' ont la même structure et que $|S| = |S'|$, $T_{max}(C) \sqsubseteq S$ et $T_{max}(C') \sqsubseteq S'$.

3.1 Définitions des équivalences statique et symbolique

Définition 3.1 (Équivalence statique $\sim$)**.** Soient S et S' deux séquences de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$ telles que $|S| = |S'|$. On dit que S et S' sont en *équivalence statique*, notée $S \sim S'$ ssi :

- $\forall \xi_1 \in \Pi, \xi_2 \in \Pi, \xi_1[S]\downarrow = \xi_2[S]\downarrow \Leftrightarrow \xi_1[S']\downarrow = \xi_2[S']\downarrow$
- $\forall \xi \in \Pi, \xi[S]\downarrow \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v) \Leftrightarrow \xi[S']\downarrow \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$

Par définition, si $|S| \neq |S'|$ alors $S \not\sim S'$.

Cette notion a été introduite par Abadi et Fournet dans [2]. Il en existe plusieurs variantes. Nous avons choisi ici de supposer que l'attaquant peut détecter si un déchiffrement réussit ou non (deuxième condition).

Définition 3.2 (Équivalence symbolique $\approx_s$)**.** Soient deux systèmes de contraintes $C = (\mathcal{C}_o, \mathcal{E}_q)$ et $C' = (\mathcal{C}'_o, \mathcal{E}'_q)$. Soient deux séquences S et S' de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$. On dit que (S, C) et (S', C') sont *en équivalence symbolique*, notée $(S, C) \approx_s (S', C')$ ssi :

- $\forall (\sigma, \xi_1, ..., \xi_n) \in Sol(C) \Rightarrow \exists \sigma'$ tq $(\sigma', \xi_1, ..., \xi_n) \in Sol(C') \wedge S\sigma \sim S'\sigma'$
- $\forall (\sigma', \xi_1, ..., \xi_n) \in Sol(C') \Rightarrow \exists \sigma$ tq $(\sigma, \xi_1, ..., \xi_n) \in Sol(C) \wedge S\sigma \sim S'\sigma'$

Si $C' = \bot$ alors $(S, C) \approx_s (S', C') \Leftrightarrow Sol(C) = \varnothing$.

Exemple 3.1. On prend les systèmes de contraintes suivant :

$$\mathcal{C}_o = \begin{cases} d, e & \Vdash & x \\ d, e, x & \Vdash & d \\ d, e, x, enc(x, f) & \Vdash & enc(enc(d, e), f) \end{cases} \qquad \mathcal{E}_q = \varnothing$$

$$S = d, e, x, enc(x, f), d$$

$$\mathcal{C}'_o = \begin{cases} a, b & \Vdash & x \\ a, b, enc(a, b) & \Vdash & a \\ a, b, enc(a, b), enc(x, h) & \Vdash & enc(enc(a, b), h) \end{cases} \qquad \mathcal{E}'_q = \varnothing$$

$$S' = a, b, enc(a, b), enc(x, h), a$$

Regardons dans un premier temps l'ensemble des solutions possibles pour $C = (\mathcal{C}_o, \mathcal{E}_q)$: Si on prend la troisième contrainte, on peut voir que la seule

manière de déduire $enc(enc(d,e),f)$ est d'utiliser un axiome sur le quatrième message car f n'est pas déductible. Donc il n'y a qu'une seule solution du premier ordre pour C :

$$\sigma = \{x \to enc(d,e)\}$$

On peut également remarquer que c'est le cas pour le système de contraintes $C' = (\mathcal{C}'_o, \mathcal{E}'_q)$. D'où la seule solution du premier ordre possible pour C' est :

$$\sigma = \{x \to enc(a,b)\}$$

Finalement, on peut remarquer qu'après application des solutions du premier ordre, les deux systèmes de contraintes sont égaux à renommage près ainsi que S et S' d'où $(S,C) \approx_s (S',C')$.

L'équivalence symbolique est facile à déterminer sur cet exemple car il n'y a en effet qu'une seule solution du premier ordre.

3.2 Support et complétude pour l'équivalence symbolique

Pour garantir l'équivalence symbolique, il n'est pas nécessaire de considérer toutes les solutions. Il suffit de se restreindre à un sous ensemble des solutions, que nous appelons *support*. Par exemple, le support des solutions en forme normale suffit pour montrer l'équivalence symbolique Nous définissons un support dans cette partie et prouvons sa complétude pour l'équivalence symbolique. Dans la suite, il sera ainsi suffisant de ne considérer que les solutions de ce support. Il est possible que l'on puisse encore se restreindre (i.e. considérer un support encore plus petit) mais nous ne chercherons pas à obtenir ici un support minimal.

Définition 3.3 (Support d'un système de contrainte)**.** Soit C un système de contraintes. Soit $\mathcal{P}$ un prédicat sur $\Sigma \times \Pi^n$. Le *support* de C pour $\mathcal{P}$, noté $\mathcal{S}_\mathcal{P}(C)$, est l'ensemble telle que :

$$\mathcal{S}_\mathcal{P}(C) = Sol(C) \cap \mathcal{P}$$

De même que pour les ensembles $Sol(C)$ et $Sol_2(C)$, on définit l'ensemble $\mathcal{S}_{2,\mathcal{P}}(C)$ tel que :

$$\mathcal{S}_{2,\mathcal{P}}(C) = \{(\xi_1,\ldots,\xi_n) \in \Pi^n \mid \exists \sigma \text{ tq } (\sigma,\xi_1,\ldots,\xi_n) \in \mathcal{S}_\mathcal{P}(C)\}$$

Définition 3.4 (Équivalence symbolique restreinte $\approx_s^{\mathcal{S}_\mathcal{P}}$)**.** Soient C et C' deux systèmes de contraintes. Soient deux séquences S et S' de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$. Soit $\mathcal{P}$ un prédicat. (S,C) et (S',C') sont *en équivalence symbolique restreinte au support* $\mathcal{S}_\mathcal{P}$, noté $(S,C) \approx_s^{\mathcal{S}_\mathcal{P}} (S',C')$, ssi :

– $\forall(\sigma,\xi_1,...,\xi_n) \in \mathcal{S}_\mathcal{P}(C) \Rightarrow \exists\sigma' \text{ tq } (\sigma',\xi_1,...,\xi_n) \in \mathcal{S}_\mathcal{P}(C') \wedge S\sigma \sim S'\sigma'$
– $\forall(\sigma',\xi_1,...,\xi_n) \in \mathcal{S}_\mathcal{P}(C') \Rightarrow \exists\sigma \text{ tq } (\sigma,\xi_1,...,\xi_n) \in \mathcal{S}_\mathcal{P}(C) \wedge S\sigma \sim S'\sigma'$

Définition 3.5 (Complétude d'un support pour $\approx_s$)**.** Soit $\mathcal{P}$ un prédicat, on dit que le support $\mathcal{S}_\mathcal{P}$ est *complet pour l'équivalence symbolique* $\approx_s$ ssi pour tous systèmes de contraintes C et C' et toutes séquences S et S',

$$(S,C) \approx_s (S',C') \Leftrightarrow (S,C) \approx_s^{\mathcal{S}_\mathcal{P}} (S',C')$$

On notera $\mathscr{S}_{comp}$ l'ensemble des supports $\mathcal{S}_\mathcal{P}$ complets pour l'équivalence symbolique $\approx_s$.

Nous avions étudié plusieurs supports complets pour l'équivalence symbolique (voir annexe B.1) Ici nous utiliserons le support, noté $\mathcal{S}_{\mathcal{P}_{un}}(C)$, qui contraint d'utiliser la même preuve de t pour toute occurrence de t.

Définition 3.6 (Prédicat $\mathcal{P}_{un}$)**.** Soit $C = (\mathcal{C}_o, \mathcal{E}_q)$ un système de contraintes. Le prédicat $\mathcal{P}_{un}$ est défini par : $\forall(\sigma,\xi_1,...,\xi_n) \in Sol(C)$

$$\mathcal{P}_{un} : \forall i,j \ \in [1..n]^2, \forall\xi \in St(\xi_i), \forall\xi' \in St(\xi_j), \xi[T_i\sigma]\downarrow = \xi'[T_j\sigma]\downarrow \Rightarrow \xi = \xi'$$

Exemple 3.2. Prenons le système de contraintes C suivant :

$$\begin{array}{lcl} a,b & \Vdash & x \\ a,b,c,enc(x,c),enc(d,a) & \Vdash & d \end{array} \qquad \mathcal{E}_q = \{z =< x,a >\}$$

Voyons maintenant une solution possible de C :

$$\sigma = \{x \to a; z \to < a,a >\}, \xi_1 = ax_1, \xi_2 = dec(ax_5, dec(ax_4, ax_3))$$

On a $(\sigma,\xi_1,\xi_2) \notin \mathcal{S}_{\mathcal{P}_{un}}(C)$. En effet, si on prend :
– $\xi_a\lfloor\_\rfloor = \_$
– $\xi_b = ax_1$
– $\xi_c\lfloor\_\rfloor = dec(ax_5,\_)$
– $\xi_d = dec(ax_4, ax_3)$
On a bien $\xi_1 = \xi_a\lfloor\xi_b\rfloor$ et $\xi_2 = \xi_c\lfloor\xi_d\rfloor$. Or $\xi_b[T_1\sigma]\downarrow = a$ et $\xi_d[T_2\sigma]\downarrow = a$, pourtant $\xi_b \neq \xi_d$. Conclusion : $(\sigma,\xi_1,\xi_2) \notin \mathcal{S}_{\mathcal{P}_{un}}(C)$

Théorème 3.7 (Complétude de $\mathcal{S}_{\mathcal{P}_{un}}$)**.** *Pour des systèmes de contraintes sans annotation,* $\mathcal{S}_{\mathcal{P}_{un}} \in \mathscr{S}_{comp}$.

Le preuve est donnée dans l'annexe C.1. Ce théorème n'est plus vrai quand on considère des annotations. Prenons l'exemple suivant :

Exemple 3.3. Soit $C = (\mathcal{C}_o, \mathcal{E}_q)$ et $C' = (\mathcal{C}'_o, \mathcal{E}'_q)$ les deux systèmes de contraintes suivants, et supposons $S = T_{max}(C)$ et $S' = T_{max}(C')$:

$$\mathcal{C}_o = \begin{array}{lll} \langle a, b\rangle & \Vdash & \langle a, b\rangle \\ \langle a, b\rangle, a, b & \Vdash & \langle a, b\rangle \end{array} \qquad \mathcal{E}_q = \varnothing$$

$$\mathcal{C}'_o = \begin{array}{lll} \langle a, b\rangle & \Vdash & \langle a, b\rangle \\ \langle a, b\rangle, a^{NoAx(a)}, b & \Vdash & \langle a, b\rangle \end{array} \qquad \mathcal{E}'_q = \varnothing$$

La seule solution possible de $\mathcal{S}_{\mathcal{P}_{un}}(C)$ est (Id, ax_1, ax_1). Il en va de même pour $\mathcal{S}_{\mathcal{P}_{un}}(C') = \{(Id, ax_1, ax_1)\}$. Ainsi a donc que $(S, C) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S', C')$. Pourtant, on a $(Id, ax_1, \langle ax_2, ax_3\rangle) \in Sol(C)$ et $(Id, ax_1, \langle ax_2, ax_3\rangle) \notin Sol(C')$ car l'annotation $NoAx(a)$ empêche l'utilisation de l'axiome ax_2 d'où $(S, C) \not\approx_s (S', C')$.

4 Règles de réduction de contrainte

L'objectif de cette partie est de montrer que l'on peut calculer, à partir de $(S, C), (S', C')$ un ensemble fini de de couples $(S_1, C_1), \ldots, (S_n, C_n), (S'_1, C'_1), \ldots, (S'_n, C'_n)$ tels que tous les C_i et C'_i sont en forme résolue et tels que $(S, C) \approx_s (S', C')$ ssi $\forall i, (S_i, C_i) \approx_s (S'_i, C'_i)$. Pour atteindre cet objectif, nous définissons des règles de transformation qui vont décomposer le problème $(S, C) \approx_s (S', C')$ en un ou plusieurs problèmes plus simples. Les règles de transformations consistent simplement en une analyse par cas (bien choisie) suivie d'une simplification. La section suivante va décrire la forme de chacune des règles que nous avons choisies ainsi que les conditions sur les systèmes de contraintes qu'elles engendrent.

4.1 Généralités sur les règles de réduction

Comme on étudie l'équivalence symbolique entre deux systèmes de contraintes, chaque règle de transformation prendra en argument deux systèmes de contraintes, notés C et C', ainsi que deux séquences de termes, notées S et S'. L'application d'une règle renvoie au maximum deux couples de système de contraintes/séquence de termes. On notera l'application de la règle R_j sur $((S, C), (S', C'))$:

$$((S, C), (S', C')) \rightsquigarrow^{R_j} \{((S_1, C_1), (S'_1, C'_1)); ((S_2, C_2), (S'_2, C'_2))\}$$

$((S_1, C_1), (S'_1, C'_1))$ et $((S_2, C_2), (S'_2, C'_2))$ sont les résultats de l'application de la règle (il se peut qu'il n'y ait pas de $((S_2, C_2), (S'_2, C'_2))$).

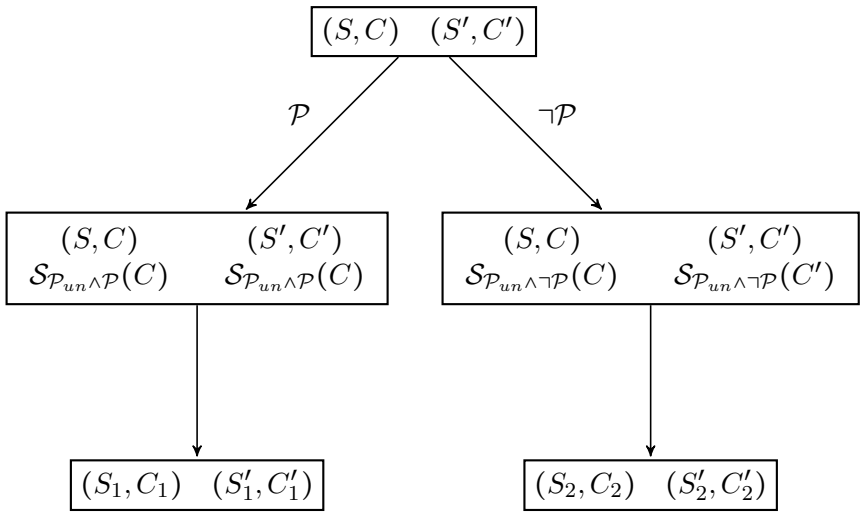


FIGURE 1 – Structure des règles : Partition par prédicats et simplifications

Une règle est définie par un prédicat $\mathcal{P}$ qui va créer une partition des supports $\mathcal{S}_{\mathcal{P}_{un}}(C)$ et $\mathcal{S}_{\mathcal{P}_{un}}(C')$:

$$\mathcal{S}_{\mathcal{P}_{un}}(C) = \mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C) \uplus \mathcal{S}_{\mathcal{P}_{un} \wedge \neg\mathcal{P}}(C)$$
$$\mathcal{S}_{\mathcal{P}_{un}}(C') = \mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C') \uplus \mathcal{S}_{\mathcal{P}_{un} \wedge \neg\mathcal{P}}(C')$$

Il s'agit d'une analyse par cas, qui est combinée avec une simplification des couples système de contraintes/séquence de termes. Le schéma de la figure 1 représente la structure générale d'une règle : Analyse par cas + simplifications.

La partition par prédicats n'est pas obligatoire. Dans ce cas, il n'y a qu'une simplification ; cela revient en fait à prendre comme prédicat $\mathcal{P} :$ true et $C_2 = C'_2 = \bot$.

Définition 4.1 (Règles adéquates)**.** Soient C, C' deux systèmes de contraintes et soient S, S' deux séquences de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$. Une règle de réduction de contraintes R est dite *adéquate* par rapport à $(S, C), (S', C')$ si elle suit le modèle de la figure 1 et si :

- Toutes les solutions des partitions $\mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C), \mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}'}(C')$ se retrouvent dans les solutions des fils $\mathcal{S}_{\mathcal{P}_{un}}(C_1), \mathcal{S}_{\mathcal{P}_{un}}(C'_1)$.
- Toutes les solutions des partitions $\mathcal{S}_{\mathcal{P}_{un} \wedge \neg\mathcal{P}}(C)$, $\mathcal{S}_{\mathcal{P}_{un} \wedge \neg\mathcal{P}'}(C')$ se retrouvent dans les solutions des fils $\mathcal{S}_{\mathcal{P}_{un}}(C_2), \mathcal{S}_{\mathcal{P}_{un}}(C'_2)$.

Théorème 4.2 (Complétude et correction des règles de réduction)**.** *Soient C et C' deux systèmes de contraintes et soient S et S' deux séquences de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$. Soit R une règle de réduction de contraintes adéquate par rapport à $(S,C),(S',C')$. Alors, pour tout $((S,C),(S',C')) \rightsquigarrow^R \{((S_1,C_1),(S'_1,C'_1));((S_2,C_2),(S'_2,C'_2))\}$,*

$$(S,C) \approx_s^{\mathcal{S}_{\mathcal{P}un}} (S',C') \Leftrightarrow \begin{cases} (S_1,C_1) \approx_s^{\mathcal{S}_{\mathcal{P}un}} (S'_1,C'_1) \\ (S_2,C_2) \approx_s^{\mathcal{S}_{\mathcal{P}un}} (S'_2,C'_2) \end{cases}$$

4.2 Définition des règles de réductions

4.2.1 Les doubles règles

Les *doubles règles* sont les règles qui engendrent les mêmes transformations sur (S,C) et (S',C'). Pour éviter les redondances dans la définition, on ne les définira uniquement sur un couple (S,C).

$$R_1 : \{T_i \Vdash f(t_1,t_2) \quad \begin{matrix} \overset{\mathcal{P}}{\nearrow} \begin{cases} T_i \Vdash t_1 \\ T_i \Vdash t_2 \end{cases} \\ \overset{\neg\mathcal{P}}{\searrow} \{T_i \Vdash f(t_1,t_2) \end{matrix}$$

$$R_2 : \begin{cases} \mathcal{C}_0 \\ T_1, v, T_2 \Vdash u \\ \mathcal{C}_1 \end{cases} \quad \alpha = mgu(u,v) \quad \begin{matrix} \overset{\mathcal{P}}{\nearrow} \begin{cases} \mathcal{C}_0\alpha \\ \mathcal{C}_1\alpha \end{cases} \\ \overset{\neg\mathcal{P}}{\searrow} \begin{cases} \mathcal{C}_0 \\ T_1, v, T_2 \Vdash u \\ \mathcal{C}_1 \end{cases} \end{matrix}$$

$$R_3 : \{T_0, \{v_1\}_{v_2}, T_2 \Vdash u_1 \quad \begin{matrix} \overset{\mathcal{P}}{\nearrow} \begin{cases} T_0, \{v_1\}_{v_2}, T_2 \Vdash v_2 \\ T_0, \{v_1\}_{v_2}, T_2, v_1 \Vdash u_1 \end{cases} \\ \overset{\neg\mathcal{P}}{\searrow} \{T_0, \{v_1\}_{v_2}, T_2 \Vdash u_1 \end{matrix}$$

$$R_4 : \{T_1, \langle v_1, v_2 \rangle, T_2 \Vdash u_1 \quad \longrightarrow \quad \{T_1, \langle v_1, v_2 \rangle, v_1, v_2, T_2 \Vdash u_1$$

FIGURE 2 – Doubles règles

Pour ces règles, dans l'une des branches, on retrouve le système de départ. Bien entendu, les annotations et les conditions d'applications (que nous ne

détaillons pas dans la figure 2) évitent les boucles (voir les figures 5 et 6 en annexe). Néanmoins, on peut aisément donner l'intuition du placement des annotations. Dans le cas de la règle R_1, on a séparé les solutions qui sont construites grâce au constructeur f. Ainsi on rajoutera l'annotation $NoCons(f(t_1, t_2))$ sur le deuxième système de contraintes pour éviter les boucles et on demandera à ce que l'annotation $NoCons(f(t_1, t_2))$ ne soit pas déjà présente pour appliquer la règle.

Sur la règle R_2, on a séparé les preuves qui utilisent l'axiome sur v pour démontrer u. Ainsi, sur le deuxième système de contraintes, on rajoutera l'annotation $NoAx(u)$ sur le terme v.

La règle R_3 est légèrement plus complexe. Ici on a séparé les preuves qui déchiffrent le terme $\{v_1\}_{v_2}$. Ainsi dans le premier système de contraintes, on demande à ce que v_2 soit déductible et l'on rajoute v_1 dans les connaissances de l'intrus. Pour éviter, les boucles, on va rajouter l'annotation $NoDec(\{v_1\}_{v_2})$. Ainsi, on effectue une seule fois cette opération. Au niveau du second système de contrainte, on va également rajouter l'annotation $NoDec(\{v_1\}_{v_2})$ mais uniquement sur cette seule contrainte. En effet, on peut toujours supposer que $\{v_1\}_{v_2}$ ne sera pas déchiffrable avec la connaissance T_0 et T_2 mais il est possible que les contraintes suivantes pourront le déchiffrer.

Enfin, la règle R_4 est similaire à la règle R_3 : On a séparé les preuves qui casser la paire $\langle v_1, v_2 \rangle$. Ainsi dans le premier système de contrainte, on rajoute v_1 et v_2 dans les connaissances de l'intrus et on rajoute l'annotation $NoPi(\langle v_1, v_2 \rangle)$ au système de contrainte. Sur le deuxième système de contrainte, on rajoutera également l'annotation $NoPi(\langle v_1, v_2 \rangle)$. Ainsi, pour appliquer la règle R_4, il faudra en plus que l'annotation $NoPi(\langle v_1, v_2 \rangle)$ ne soit pas déjà présente.

On peut remarquer que grâce aux règles R_3 et R_4, on tend à obtenir des systèmes de contraintes dans lesquelles les preuves ne contiennent que des axiomes et des constructeurs (les destructeurs ayant été devinés au préalable).

4.2.2 Les règles non-symétriques

Ces règles traitent des cas où les termes apparaissent à des positions identiques dans les deux systèmes de contraintes n'ont pas la même forme :

- ou bien il n'est pas possible pour l'attaquant de faire la différence et dans ce cas, les systèmes restent inchangés.
- ou bien il peut faire la différence et dans ce cas, les systèmes ne doivent pas avoir de solution.

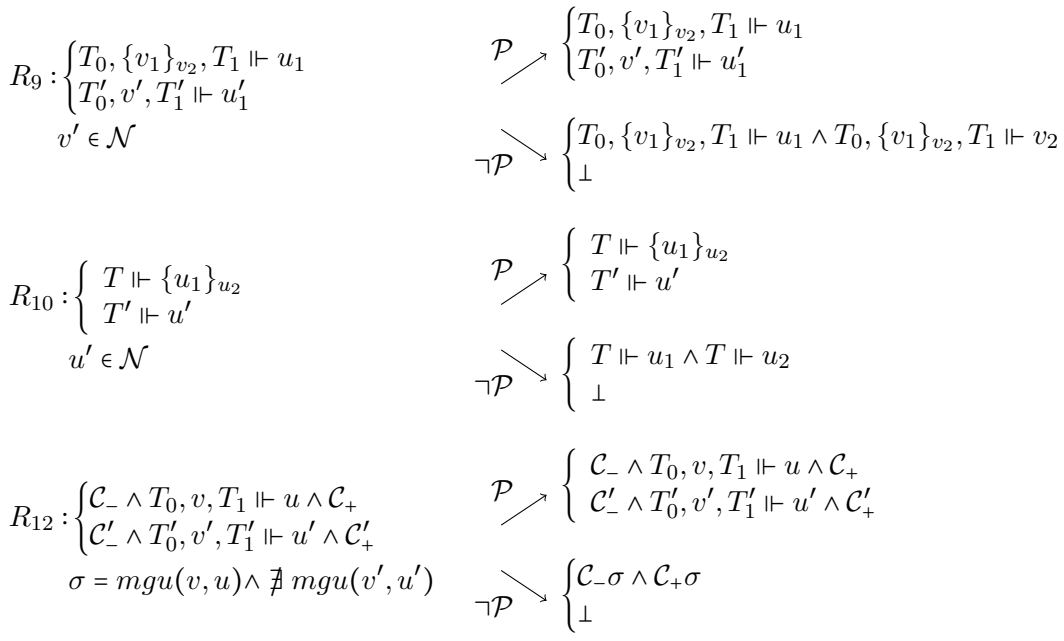


FIGURE 3 – La règle non-symétrique

La règle R_{11} n'est pas donnée dans la figure 3 car elle ne porte uniquement que les annotations.

4.2.3 Les règles de gestion des variables

Ces règles traitent des cas où, à des positions identiques dans les deux systèmes, d'un coté, on trouve un terme et de l'autre une variable. Il s'agit du cas le plus délicat.

La règle R_{13} traite du cas où x a deux occurences dans les membres droits et n'est pas décrit ici.

La règle R_{14} considère les cas où une variable x apparait à la fois à gauche et à droite dans un système de contraintes. Aux positions correspondantes de l'autre système, les termes doivent être identiquses. De plus, pour toute solution θ, $x\theta$ admet une preuve et peut donc être reconstruite à partir du reste du membre gauche : x est inutile dans le membre gauche.

Même après application de la règle R_{14}, une variable x peut apparaître à droite dans l'un des systèmes alors qu'à la position correspondante dans l'autre système, se trouve un terme non variable $f(u_1', u_2')$. Dans ce cas,

$$R_{14} : \begin{cases} T_1 \Vdash x \wedge T_2, x, T_3 \Vdash u_2 \\ T_1' \Vdash u_1' \wedge T_2', u, T_3' \Vdash u_2' \\ x \in \mathcal{X}_v, \sigma = mgu(u_1', u) \end{cases} \quad \begin{array}{l} \mathcal{P} \nearrow \begin{cases} T_1 \Vdash x \wedge T_2, T_3 \Vdash u_2 \\ T_1'\sigma \Vdash u_1'\sigma \wedge T_2'\sigma, T_3'\sigma \Vdash u_2'\sigma \end{cases} \\ \neg\mathcal{P} \searrow \begin{cases} \bot \\ C', \mathcal{E}_q' \cup \{u \neq u_1'\}, S' \end{cases} \end{array}$$

$$R_{16} : \begin{cases} T \Vdash x \\ T' \Vdash f(u_1', u_2') \\ x \in \mathcal{X}_v \wedge \sigma = \{x \to f(x_a, x_b)\} \\ \wedge\, x_a, x_b \text{ variables fraiches} \end{cases} \quad \begin{array}{l} \mathcal{P} \nearrow \begin{cases} T \Vdash x \\ T' \Vdash f(u_1', u_2') \end{cases} \\ \neg\mathcal{P} \searrow \begin{cases} T \Vdash x_a \wedge T \Vdash x_b \\ T' \Vdash u_1' \wedge T' \Vdash u_2' \end{cases} \end{array}$$

$$R_{17} : \begin{cases} T \Vdash f(u_1, u_2) \\ T' \Vdash x \\ x \in \mathcal{X}_v \wedge \sigma = \{x \to f(x_a, x_b)\} \\ \wedge\, x_a, x_b \text{ variables fraiches} \\ \wedge \text{ aucune règle n'est applicable} \end{cases} \quad \begin{array}{l} \mathcal{P} \nearrow \begin{cases} T \Vdash f(u_1, u_2) \\ T' \Vdash x \end{cases} \\ \neg\mathcal{P} \searrow \begin{cases} T \Vdash u_1 \wedge T \Vdash u_2 \\ T' \Vdash x_a \wedge T' \Vdash x_b \end{cases} \end{array}$$

FIGURE 4 – La règle de gestion de variables

ou bien $f(u_1', u_2')$ ne peut pas être obtenu par construction et dans ce cas, les systèmes restent inchangés. Ou bien, x doit pouvoir être obtenue par construction. Dans ce cas, une solution θ doit être telle que $x\theta = f(u_1, u_2)$. On introduit alors deux variable fraîches x_a, x_b telle que $x = f(x_a, x_b)$. Cette règle fait croître le nombre de variables ce qui pourrait poser un problème de terminaison. Pour cette raison, R_{16} ne rajoute de variables qu'à l'un des systèmes de contraintes (le premier dans la règle R_{16}). La règle R_{17} qui rajoute des variables dans le deuxième système de contraintes ne s'applique qu'en dernier ressort, ceci permet d'assurer la terminaison car l'application de la règle R_{17} n'introduit pas d'occurence de la règle R_{16}. Par conséquent, le nombre de variables introduites reste borné.

4.2.4 Les règles d'échec

Quand aucune autre règle n'est applicable et que l'un des deux systèmes n'est pas en forme résolue, une règle d'échec s'applique : pour être équivalent, les deux systèmes doivent être insatisfaisables.

4.3 Terminaison des règles de réduction

Les règles énoncées en annexe A.2 possède l'ensemble des propriétés souhaitées mais les preuves de certaines d'entre-elles ne sont pas achevées. On énonce donc ici quelques conjectures.

Conjecture 4.3 (Terminaison)**.** *L'ensemble des règles de réduction terminent : il n'y a pas de séquence infinie* $[(S,C),(S',C')] \rightsquigarrow^{R_{i_1}} \cdots \rightsquigarrow^{R_{i_n}} [(S_n,C_n),(S'_n,C'_n)] \rightsquigarrow^{R_{i_n}} \ldots$.

Conjecture 4.4 (Etat terminal des systèmes de contraintes)**.** *Soit* C,C' *deux systèmes de contraintes et* S,S' *deux séquences de termes tels que* C *et* C' *ne contiennent aucune annotation. Soit* C_1,C'_1 *et* S_1,S'_1 *tels que* $((S,C),(S',C')) \rightsquigarrow^{R_{j_1}} \cdots \rightsquigarrow^{R_{j_n}} ((S_1,C_1),(S'_1,C'_1))$ *et aucune règle n'est applicable sur* $((S_1,C_1),(S'_1,C'_1))$*. Alors :*

- *ou bien* $(S_1,C_1) = \bot$ *et* $(S'_1,C'_1) = \bot$
- *ou bien* $(S_1,C_1) = \bot$ *et* C'_1 *est en forme résolue*
- *ou bien* $(S'_1,C'_1) = \bot$ *et* C_1 *est en forme résolue*
- *ou bien* C_1 *et* C'_1 *sont tous les deux en forme résolue.*

Comme conséquence des théorèmes 3.7, 4.2 et des conjectures 4.3, 4.4. On obtient le corollaire suivant :

Corollaire 4.5 (Principe de l'algorithme)**.** *Soit* C,C' *deux systèmes de contraintes et* S,S' *deux séquences de termes tels que* C *et* C' *ne contiennent aucune annotation.* $(S,C) \approx_s (S',C')$ *ssi il existe une séquence d'indices* $\{i_j\}_{j\in[1..m]}$ *telle que :*

$$\begin{bmatrix}(S,C)\\(S',C')\end{bmatrix} \rightsquigarrow^{R_{i_1}} \cdots \rightsquigarrow^{R_{i_m}} \left\{\begin{bmatrix}(S_1,C_1)\\(S'_1,C'_1)\end{bmatrix},\ldots,\begin{bmatrix}(S_n,C_n)\\(S'_n,C'_n)\end{bmatrix}\right\}$$

et aucune règle n'est applicable à l'un des $\begin{bmatrix}(S_k,C_k)\\(S'_k,C'_k)\end{bmatrix}$ *et* $\forall k$*,*

- *ou bien* $C_k = C'_k = \bot$
- *ou bien* C_k *et* C'_k *sont en forme résolue et* $(S_k,C_k) \approx_s (S'_k,C'_k)$*.*

Références

[1] Spore, the security protocol open repository. `//www.lsv.ens-cachan.fr/spore`.

[2] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. *SIGPLAN Not.*, 36(3) :104–115, 2001.

[3] R. M. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theor. Comput. Sci.*, 290(1) :695–740, 2003.

[4] M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc. 12th ACM Conference on Computer and Communications Security (CCS'05)*, Alexandria, Virginia, USA, 2005. ACM Press.

[5] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. 14th IEEEComputer Security Foundations Workshop (CSFW-14)*, Cape Breton, Nova Scotia, Canada, 2001.

[6] B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1) :3–51, Feb.–Mar. 2008.

[7] Y. Chevalier and M. Rusinowitch. Combining Intruder Theories. In *Proc. ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*. Springer, 2005.

[8] H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*, pages 109–118, Alexandria, Virginia, USA, Oct. 2008. ACM Press.

[9] H. Comon-Lundh, V. Cortier, and E. Zalinescu. Deciding security properties of cryptographic protocols. application to key cycles. *Transaction on Computational Logic*, 2009. To appear. A preliminary verion is available at `http://arxiv.org/abs/0708.3564`.

[10] V. Cortier and S. Delaune. A method for proving observational equivalence. In *Proc. 22nd IEEEComputer Security Foundations Symposium (CSF'09)*. Comp. Soc. Press, 2009. To appear.

[11] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13(3) :423–482, 2005.

[12] D. Dolev and A. C. Yao. On the security of public key protocols. Technical report, Stanford, CA, USA, 1981.

[13] H. Huttel. Deciding framed bisimulation. In *4th International Workshop on Verification of Infinite State Systems INFINITY'02*, pages 1–20, 2002.

[14] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security*, 2001.

[15] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, 2001.

[16] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*. Comp. Soc. Press, 2001.

A Définitions des règles

A.1 Formalisme

Pour représenter les règles, on utilise l'appariement au niveau de détail approprié et donc on n'indique pas les composants qui restent inchangés dans la règle. Cela évite de donner des détails sans importances qui risquent de rendre difficile la compréhension de la règle.

Nous utiliserons également deux formalismes pour représenter les couples $((S,C),(S',C'))$. On sait que pour chaque couple que l'on considère, $C = (\mathcal{C}_o, \mathcal{E}_q)$ et $C' = (\mathcal{C}'_o, \mathcal{E}'_q)$ ont la même structure et $|S| = |S'|$. Ainsi, on représentera $((S,C),(S',C'))$ comme suit :

$$\left(\begin{bmatrix} C_o \\ C'_o \end{bmatrix}, \begin{bmatrix} \mathcal{E}_q \\ \mathcal{E}'_q \end{bmatrix}, \begin{bmatrix} S \\ S' \end{bmatrix}\right) \stackrel{def}{=} ((S,C),(S',C'))$$

$$\begin{bmatrix} S \\ S' \end{bmatrix} \stackrel{def}{=} \begin{bmatrix} t_1 \\ t'_1 \end{bmatrix}, \ldots, \begin{bmatrix} t_n \\ t'_n \end{bmatrix} \quad \text{si} \quad \begin{array}{l} S = [t_1, \ldots, t_n] \\ S' = [t'_1, \ldots, t'_n] \end{array}$$

$$\begin{bmatrix} \mathcal{C}_o \\ \mathcal{C}'_o \end{bmatrix} \stackrel{def}{=} \begin{cases} \begin{bmatrix} T_1 \\ T'_1 \end{bmatrix} \Vdash \begin{bmatrix} u_1 \\ u'_1 \end{bmatrix} \\ \ldots \\ \begin{bmatrix} T_n \\ T'_n \end{bmatrix} \Vdash \begin{bmatrix} u_n \\ u'_n \end{bmatrix} \end{cases} \quad \text{si} \quad \begin{array}{l} \mathcal{C}_o = [T_1 \Vdash u_1, \ldots, T_n \Vdash u_n] \\ \mathcal{C}'_o = [T'_1 \Vdash u'_1, \ldots, T'_n \Vdash u'_n] \end{array}$$

$$\begin{bmatrix} T_i \\ T'_i \end{bmatrix} \stackrel{def}{=} \begin{bmatrix} t_1 \\ t'_1 \end{bmatrix}, \ldots, \begin{bmatrix} t_m \\ t'_m \end{bmatrix} \quad \text{si} \quad \begin{array}{l} T_i = [t_1, \ldots, t_m] \\ T'_i = [t'_1, \ldots, t'_m] \end{array}$$

Ce formalisme est intéressant pour identifier les deux termes de C qui sont à la même position p sur la même contrainte i par exemple. Bien que cela n'est pas le cas ci-dessus, il est possible de rajouter les ensembles d'annotations sur les termes comme expliqué dans la remarque 1. On aurait par exemple :

$$\begin{bmatrix} T_i \\ T'_i \end{bmatrix} \stackrel{def}{=} \begin{bmatrix} t_1^{F_1} \\ t'_1{}^{F'_1} \end{bmatrix}, \ldots, \begin{bmatrix} t_m^{F_m} \\ t'_m{}^{F'_m} \end{bmatrix}$$

Néanmoins, il est possible que nous ne nous intéressions pas toujours aux

deux valeurs d'un couple. Dans ce cas, on utilisera le formalisme suivant :

$$\left(\overline{\mathcal{C}_o}, \overline{\mathcal{E}_q}, \overline{S}\right) \stackrel{def}{=} ((S, C), (S', C'))$$

$$\overline{S} \stackrel{def}{=} \overline{t_1}, \ldots, \overline{t_n}$$

$$\overline{t_i} \stackrel{def}{=} \begin{bmatrix} t_1 \\ t'_1 \end{bmatrix}$$

$$\overline{\mathcal{C}_o} \stackrel{def}{=} \begin{cases} \overline{T_1} \Vdash \overline{u_1} \\ \ldots \\ \overline{T_n} \Vdash \overline{u_n} \end{cases}$$

$$\text{ou bien } \overline{\mathcal{C}_o} \stackrel{def}{=} \begin{cases} \overline{\mathcal{C}_-} \\ \overline{T_i} \Vdash \overline{u_i} \\ \ldots \\ \overline{T_j} \Vdash \overline{u_j} \\ \overline{\mathcal{C}_+} \end{cases}$$

où $\overline{\mathcal{C}_-}$ représente toutes les contraintes avant $\overline{T_i} \Vdash \overline{u_i}$
et $\overline{\mathcal{C}_+}$ représente toutes les contraintes après $\overline{T_j} \Vdash \overline{u_j}$

On définit deux opérateurs *fst* et *snd* tels que si $\overline{H} = \begin{bmatrix} H \\ H' \end{bmatrix}$, alors $fst(\overline{H}) = H$ et $snd(\overline{H}) = H'$.

On introduit également des notations pour les opérations sur les annotations :

- Si t^F est un terme, alors $t^{F+NoAx(u)} \stackrel{def}{=} t^{F \cup \{NoAx(u)\}}$ (Valable pour les autres annotations)
- Si $T = [t_1^{F_1}, \ldots, t_n^{F_n}]$ est une séquence, alors $T^{SF+NoAx(u)} \stackrel{def}{=} [t_1^{F_1+NoAx(u)}, \ldots, t_n^{F_n+NoAx(u)}]$.
- Si $T = [t_1^{F_1}, \ldots, t_n^{F_n}]$ est une séquence, alors $T^{SF+NoAx(u)_{|\neq p}} \stackrel{def}{=} [t_1^{F_1+NoAx(u)}, \ldots, t_{p-1}^{F_{p-1}+NoAx(u)}, t_p^{F_p}, t_{p+1}^{F_{p+1}+NoAx(u)}, \ldots, t_n^{F_n+NoAx(u)}]$
- Si $T = [t_1^{F_1}, \ldots, t_n^{F_n}]$ est une séquence, alors $T^{SF+NoAx(u)_{|p}} \stackrel{def}{=} [t_1^{F_1}, \ldots, t_{p-1}^{F_{p-1}}, t_p^{F_p+NoAx(u)}, t_{p+1}^{F_{p+1}}, \ldots, t_n^{F_n}]$.

Ces deux formalismes ainsi que ces opérations sur les annotations pourront être mélangées à loisir comme le montre l'exemple suivant de la règle R_{10} :

Exemple A.1.

$$R_{10} : \begin{cases} \overline{T} \Vdash \begin{bmatrix} \{u_1\}_{u_2}{}^F \\ u' \end{bmatrix} \\ \begin{bmatrix} \mathcal{E}_q \\ \mathcal{E}'_q \end{bmatrix} \end{cases} \quad u' \in \mathcal{N} \wedge NoCons(\{u_1\}_{u_2}) \notin F$$

$$F_1 \nearrow \begin{cases} \begin{bmatrix} fst(\overline{T}) \Vdash \{u_1\}^F_{u_2} \\ \bot \end{bmatrix} \\ \begin{bmatrix} fst(\overline{T}) \Vdash u_2^F \\ \bot \end{bmatrix} \\ \begin{bmatrix} \mathcal{E}_q \cup \{w \neq \{u_1\}_{u_2} \mid w \in E\} \\ \bot \end{bmatrix} \end{cases}$$

$$F_2 \searrow \begin{cases} \overline{T} \Vdash \begin{bmatrix} \{u_1\}_{u_2}{}^{F+NoCons(\{u_1\}_{u_2})} \\ u' \end{bmatrix} \\ \begin{bmatrix} \mathcal{E}_q \\ \mathcal{E}'_q \end{bmatrix} \end{cases}$$

avec $E = \{w \mid NoCons(w) \in F \wedge \exists mgu(w, \{u_1\}_{u_2})\}$.

Les conditions d'applications de cette règle ne portent que sur une contrainte où seul le membre droit est important. On a utilisé le premier formalisme pour le membre droit car des conditions différentes s'appliquent sur les deux termes du couple. Sur cette règle, si $(S, C), (S', C')$ est le couple en entrée et $(S_1, C_1), (S'_1, C'_1)$ et $(S_2, C_2), (S'_2, C'_2)$ sont les couples en sortie, alors $S = S_1 = S_2$ et $S' = S'_1 = S'_2$ puisqu'ils n'apparaissent pas sur la règle.

A.2 Enoncé des règles

$$R_1 : \begin{cases} \begin{cases} T_1 \Vdash u_1^{F_1} \\ \dots \\ T_i \Vdash f(t_1,t_2)^{F_i} \\ \dots \\ T_n \Vdash u_n^{F_n} \end{cases} \\ \mathcal{E}_q \\ NoCons(f(t_1,t_2)) \notin F_i \end{cases} \quad \begin{array}{l} \nearrow^{F_1} \\ \\ \searrow_{F_2} \end{array} \quad \begin{cases} \begin{cases} T_1 \Vdash u_1^{F_1} \\ \dots \\ T_i \Vdash t_1^{F_i} \\ T_i \Vdash t_2^{F_i} \\ \dots \\ T_n \Vdash u_n^{F_n} \end{cases} \\ \mathcal{E}_q \cup \{f(v_1,v_2) \neq f(t_1,t_2) \mid f(v_1,v_2) \in E\} \end{cases}$$

$$\begin{cases} \begin{cases} T_1 \Vdash u_1^{F_1+NoCons(f(t_1,t_2))} \\ \dots \\ T_i \Vdash f(t_1,t_2)^{F_i+NoCons(f(t_1,t_2))} \\ \dots \\ T_n \Vdash u_n^{F_n+NoCons(f(t_1,t_2))} \end{cases} \\ \mathcal{E}_q \end{cases}$$

avec $E = \{f(v_1,v_2) \mid NoCons(f(v_1,v_2)) \in F_i \wedge \exists mgu(f(v_1,v_2), f(t_1,t_2))\}$

$$R_2 : \begin{cases} \begin{cases} \mathcal{C}_0^{CF_0} \\ T_1^{SF_1}, v^F, T_2^{SF_2} \Vdash u^{F'} \\ \mathcal{C}_1^{CF_1} \end{cases} \\ \mathcal{E}_q \\ S \\ \alpha = mgu(u,v) \wedge |T_1| = p-1 \\ \wedge\, NoAx(u\alpha) \notin F\alpha \end{cases} \quad \begin{array}{l} \nearrow^{F_1} \\ \\ \searrow_{F_2} \end{array} \quad \begin{cases} \begin{cases} \mathcal{C}_0^{CF_0}\alpha \\ \mathcal{C}_1^{CF_1}\alpha \end{cases} \\ \mathcal{E}_q\alpha \cup \{w\alpha \neq u\alpha \mid w \in E\} \\ S\alpha \end{cases}$$

$$\begin{cases} \begin{cases} \mathcal{C}_0^{CF_0+NoAx(u)_{|p}} \\ T_1^{SF_1}, v^{F+NoAx(u)}, T_2^{SF_2} \Vdash u^{F'} \\ \mathcal{C}_1^{CF_1+NoAx(u)_{|p}} \end{cases} \\ \mathcal{E}_q \\ S \end{cases}$$

avec $E = \{w \mid NoAx(w) \in F \wedge \exists mgu(w\alpha, u\alpha)\}$

FIGURE 5 – Règle R_1 et R_2

$$R_3 : \begin{cases} T & \Vdash u^{F_0} \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2 & \Vdash u_1^{F_1} \\ \dots & \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2 & \Vdash u_2^{F_2} \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2, T_3 & \Vdash u_3^{F_3} \\ \dots & \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2, T_n & \Vdash u_n^{F_4} \\ & \\ \mathcal{E}_q & \\ S \equiv T_1, \{v_1\}_{v_2}, T_2, S' & \end{cases}$$

$$\{[T_0, \{v_1\}_{v_2} \notin T]$$
$$\vee\, [T_0, \{v_1\}_{v_2} \sqsubseteq T$$
$$\wedge\, NoDec(\{v_1\}_{v_2}) \in F_0]\}$$
$$\wedge\, NoDec(\{v_1\}_{v_2}) \notin F_0$$

$F_1 \nearrow$

$$\begin{cases} T & \Vdash u^{F_0} \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2 & \Vdash v_2^{F_1+NoDec(\{v_1\}_{v_2})} \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2, v_1^{\varnothing} & \Vdash u_1^{F_1+NoDec(\{v_1\}_{v_2})} \\ \dots & \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2, v_1^{\varnothing} & \Vdash u_2^{F_2+NoDec(\{v_1\}_{v_2})} \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2, v_1^{\varnothing}, T_3 & \Vdash u_3^{F_3+NoDec(\{v_1\}_{v_2})} \\ \dots & \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2, v_1^{\varnothing}, T_n & \Vdash u_n^{F_4+NoDec(\{v_1\}_{v_2})} \\ & \\ \mathcal{E}_q \cup \{w \neq \{v_1\}_{v_2} \mid w \in E\} & \\ S \equiv T_1, \{v_1\}_{v_2}, T_2, v_1, S' & \end{cases}$$

$F_2 \searrow$

$$\begin{cases} T & \Vdash u^{F_0} \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2 & \Vdash u_1^{F_1+NoDec(\{v_1\}_{v_2})} \\ \dots & \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2 & \Vdash u_2^{F_2+NoDec(\{v_1\}_{v_2})} \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2, T_3 & \Vdash u_3^{F_3} \\ \dots & \\ T_0\,, \{v_1\}_{v_2}{}^F, T_2, T_n & \Vdash u_n^{F_4} \\ & \\ \mathcal{E}_q & \\ S \equiv T_1, \{v_1\}_{v_2}, T_2, S' & \end{cases}$$

avec $E = \{w \mid NoAx(w) \in F \wedge \exists mgu(w, \{v_1\}_{v_2})\}$

$$R_4 : \begin{cases} T \Vdash u^{F_0} & \\ T_1, \langle v_1, v_2\rangle^F, T_2 \Vdash u_1^{F_1} & \\ \dots & \\ T_1, \langle v_1, v_2\rangle^F, T_n \Vdash u_n^{F_n} & \\ & \\ \mathcal{E}_q & \\ S \equiv T_1, \langle v_1, v_2\rangle, S' & \end{cases} \xrightarrow{F_1} \begin{cases} T \Vdash u^{F_0+NoPi(\langle v_1,v_2\rangle)} & \\ T_1, \langle v_1, v_2\rangle^F, v_1^{\varnothing}, v_2^{\varnothing}, T_2 \Vdash u_1^{F_1+NoPi(\langle v_1,v_2\rangle)} & \\ \dots & \\ T_1, \langle v_1, v_2\rangle^F, v_1^{\varnothing}, v_2^{\varnothing}, T_n \Vdash u_n^{F_n+NoPi(\langle v_1,v_2\rangle)} & \\ & \\ \mathcal{E}_q \cup \{w \neq \langle v_1, v_2\rangle \mid w \in E\} & \\ S \equiv T_1, \langle v_1, v_2\rangle, v_1, v_2, S' & \end{cases}$$

$$NoPi(\langle v_1, v_2\rangle) \notin F_1 \wedge T_1, \langle v_1, v_2\rangle \notin T$$

avec $E = \{w \mid NoAx(w) \in F \wedge \exists mgu(w, \langle v_1, v_2\rangle)\}$

FIGURE 6 – Règle R_3 et R_4

$$R_5 : \begin{cases} \overline{T_1}, \begin{bmatrix} \langle v_1, v_2 \rangle \\ v' \end{bmatrix}, \overline{T_2} \Vdash \overline{u} \\ Top(v') \neq <> \wedge v' \notin \mathcal{X}_v \end{cases} \quad \begin{array}{l} \overset{F_1}{\nearrow} \left\{ \begin{bmatrix} \mathcal{C}_o, \mathcal{E}_q, S \\ \bot \end{bmatrix} \right. \\ \overset{F_2}{\searrow} \left\{ \begin{bmatrix} \bot \\ \mathcal{C}'_o, \mathcal{E}'_q, S' \end{bmatrix} \right. \end{array}$$

$$R_6 : \begin{cases} \overline{T} \Vdash \begin{bmatrix} \langle u_1, u_2 \rangle \\ u' \end{bmatrix} \\ Top(u') \neq <> \wedge u' \notin \mathcal{X}_v \end{cases} \quad \begin{array}{l} \overset{F_1}{\nearrow} \left\{ \begin{bmatrix} \mathcal{C}_o, \mathcal{E}_q, S \\ \bot \end{bmatrix} \right. \\ \overset{F_2}{\searrow} \left\{ \begin{bmatrix} \bot \\ \mathcal{C}'_o, \mathcal{E}'_q, S' \end{bmatrix} \right. \end{array}$$

$$R_7 : \begin{cases} \overline{\mathcal{C}_o}, \begin{bmatrix} \mathcal{E}_q \cup \{u \neq u\} \\ \mathcal{E}'_q \end{bmatrix}, \overline{S} \end{cases} \quad \overset{F_1}{\longrightarrow} \left\{ \begin{bmatrix} \bot \\ \mathcal{C}'_o, \mathcal{E}'_q, S' \end{bmatrix} \right.$$

$$R_8 : \begin{cases} \begin{bmatrix} T \\ T' \end{bmatrix} \Vdash \begin{bmatrix} u^{F_1} \\ u' \end{bmatrix} \\ \forall \{v_1\}_{v_2} \in T, NoDec(\{v_1\}_{v_2}) \in F_1 \\ \wedge \forall \langle v_1, v_2 \rangle \in T, NoPi(\langle v_1, v_2 \rangle) \in F_1 \\ \wedge \forall v^F \in T, \exists \sigma = mgu(v, u) \Rightarrow NoAx(u\sigma) \in F\sigma \\ \wedge NoCons(u) \in F_1 \\ \wedge \forall v \in T, v \notin \mathcal{X}_v \end{cases} \quad \overset{F_1}{\longrightarrow} \left\{ \begin{bmatrix} \bot \\ C', \mathcal{E}'_q, S' \end{bmatrix} \right.$$

FIGURE 7 – Les règles d'échec

$$R_9 : \begin{cases} \overline{T} \Vdash \begin{bmatrix} u_0^{F_0} \\ u_0' \end{bmatrix} \\ \overline{T_0}, \begin{bmatrix} \{v_1\}_{v_2} \\ v' \end{bmatrix}, \overline{T_1} \Vdash \begin{bmatrix} u_1^{F_1} \\ u_1' \end{bmatrix} \\ \dots \\ \overline{T_0}, \begin{bmatrix} \{v_1\}_{v_2} \\ v' \end{bmatrix}, \overline{T_n} \Vdash \begin{bmatrix} u_n^{F_n} \\ u_n' \end{bmatrix} \\ \overline{S} = \overline{T_0}, \begin{bmatrix} \{v_1\}_{v_2} \\ v' \end{bmatrix}, \overline{T_n}, \overline{S_2} \end{cases}$$
$$fst(\overline{T_0}), \{v_1\}_{v_2} \notin fst(\overline{T}) \wedge NoDec(\{v_1\}_{v_2}) \notin F_1 \wedge v' \in \mathcal{N}$$

$$F_1 \nearrow \begin{cases} \overline{T} \Vdash \begin{bmatrix} u_0^{F_0+NoDec(\{v_1\}_{v_2})} \\ u_0' \end{bmatrix} \\ \overline{T_0}, \begin{bmatrix} \{v_1\}_{v_2} \\ v' \end{bmatrix}, \overline{T_1} \Vdash \begin{bmatrix} u_1^{F_1+NoDec(\{v_1\}_{v_2})} \\ u_1' \end{bmatrix} \\ \dots \\ \overline{T_0}, \begin{bmatrix} \{v_1\}_{v_2} \\ v' \end{bmatrix}, \overline{T_n} \Vdash \begin{bmatrix} u_n^{F_n+NoDec(\{v_1\}_{v_2})} \\ u_n' \end{bmatrix} \end{cases}$$
$$F_2 \searrow \begin{cases} \overline{S} = \overline{T_0}, \begin{bmatrix} \{v_1\}_{v_2} \\ v' \end{bmatrix}, \overline{T_n}, \overline{S_2} \end{cases}$$
$$\begin{cases} \begin{bmatrix} \left\{ \begin{array}{l} fst(\overline{\mathcal{C}_o}) \\ fst(\overline{S}) \Vdash v_2^F \end{array} \right\}, fst(\overline{\mathcal{E}_q}), fst(\overline{S}) \end{bmatrix} \\ \begin{bmatrix} \\ \bot \end{bmatrix} \end{cases}$$

FIGURE 8 – La règle R_9

$$R_{10} : \begin{cases} \overline{T} \Vdash \begin{bmatrix} \{u_1\}_{u_2}{}^F \\ u' \end{bmatrix} \\ \begin{bmatrix} \mathcal{E}_q \\ \mathcal{E}_q' \end{bmatrix} \end{cases}$$
$$u' \in \mathcal{N} \wedge NoCons(\{u_1\}_{u_2}) \notin F$$

$$F_1 \nearrow \begin{cases} \begin{bmatrix} fst(\overline{T}) \Vdash u_1^F \\ \bot \end{bmatrix} \\ \begin{bmatrix} fst(\overline{T}) \Vdash u_2^F \\ \bot \end{bmatrix} \\ \begin{bmatrix} \mathcal{E}_q \cup \{w \neq \{u_1\}_{u_2} \mid w \in E\} \\ \bot \end{bmatrix} \end{cases}$$
$$F_2 \searrow \begin{cases} \overline{T} \Vdash \begin{bmatrix} \{u_1\}_{u_2}{}^{F+NoCons(\{u_1\}_{u_2})} \\ u' \end{bmatrix} \\ \begin{bmatrix} \mathcal{E}_q \\ \mathcal{E}_q' \end{bmatrix} \end{cases}$$

avec $E = \{w \mid NoCons(w) \in F \wedge \exists mgu(w, \{u_1\}_{u_2})\}$

FIGURE 9 – La règle R_{10}

$$R_{11} : \begin{cases} \overline{T} \Vdash \begin{bmatrix} f(u_1,u_2)^F \\ f(u_1',u_2')^{F'} \end{bmatrix} \\ \begin{bmatrix} \mathcal{E}_q \\ \mathcal{E}_q' \end{bmatrix} \\ NoCons(f(u_1,u_2)) \notin F \\ \wedge\, NoCons(f(u_1',u_2')) \in F' \end{cases} \quad \begin{array}{l} \nearrow^{F_1} \\ \searrow_{F_2} \end{array} \quad \begin{cases} \begin{bmatrix} fst(\overline{T}) \Vdash u_1^F \\ \bot \end{bmatrix} \\ \begin{bmatrix} fst(\overline{T}) \Vdash u_2^F \\ \bot \end{bmatrix} \\ \begin{bmatrix} \mathcal{E}_q \cup \{w \neq f(u_1,u_2) \mid w \in E\} \\ \bot \end{bmatrix} \end{cases} \begin{cases} \overline{T} \Vdash \begin{bmatrix} f(u_1,u_2)^{F+NoCons(f(u_1,u_2))} \\ f(u_1',u_2')^{F'} \end{bmatrix} \\ \begin{bmatrix} \mathcal{E}_q \\ \mathcal{E}_q' \end{bmatrix} \end{cases}$$

avec $E = \{w \mid NoCons(w) \in F \wedge \exists mgu(w, f(u_1,u_2))\}$

FIGURE 10 – La règle R_{11}

$$R_{12} : \begin{cases} \begin{cases} \overline{\mathcal{C}_-} \\ \overline{T_1}, \begin{bmatrix} v^{F_2} \\ v'^{F_2'} \end{bmatrix}, \overline{T_3} \Vdash \begin{bmatrix} u \\ u' \end{bmatrix} \\ \overline{\mathcal{C}_+} \end{cases} \\ \begin{bmatrix} \mathcal{E}_q \\ \mathcal{E}_q' \end{bmatrix}, \overline{S} \\ \sigma = mgu(v,u) \wedge NoAx(u\sigma) \notin F_2\sigma \\ \wedge \begin{cases} \nexists\, mgu(v',u') \\ \vee \begin{cases} \sigma' = mgu(v',u') \\ \wedge NoAx(u'\sigma') \in F_2'\sigma' \end{cases} \end{cases} \end{cases} \quad \begin{array}{l} \nearrow^{F_1} \\ \searrow_{F_2} \end{array} \quad \begin{cases} \begin{cases} \begin{bmatrix} fst(\overline{\mathcal{C}_-})\sigma \\ \bot \end{bmatrix} \\ \begin{bmatrix} fst(\overline{\mathcal{C}_+})\sigma \\ \bot \end{bmatrix} \end{cases} \\ \begin{bmatrix} \mathcal{E}_q \cup \{w\alpha \neq u\alpha \mid w \in E\} \\ \bot \end{bmatrix} \\ \begin{bmatrix} fst(S)\sigma \\ \bot \end{bmatrix} \end{cases} \begin{cases} \begin{cases} \overline{\mathcal{C}_-} \\ \overline{T_1^{F_1}}, \begin{bmatrix} v^{F_2+NoAx(u)} \\ v'^{F_2'} \end{bmatrix}, \overline{T_3^{F_3}} \Vdash \begin{bmatrix} u \\ u' \end{bmatrix} \\ \overline{\mathcal{C}_+} \end{cases} \\ \begin{bmatrix} \mathcal{E}_q \\ \mathcal{E}_q' \end{bmatrix}, \overline{S} \end{cases}$$

avec $E = \{w \mid NoAx(w) \in F \wedge \exists mgu(w\alpha, u\alpha)\}$

FIGURE 11 – La règle R_{12}

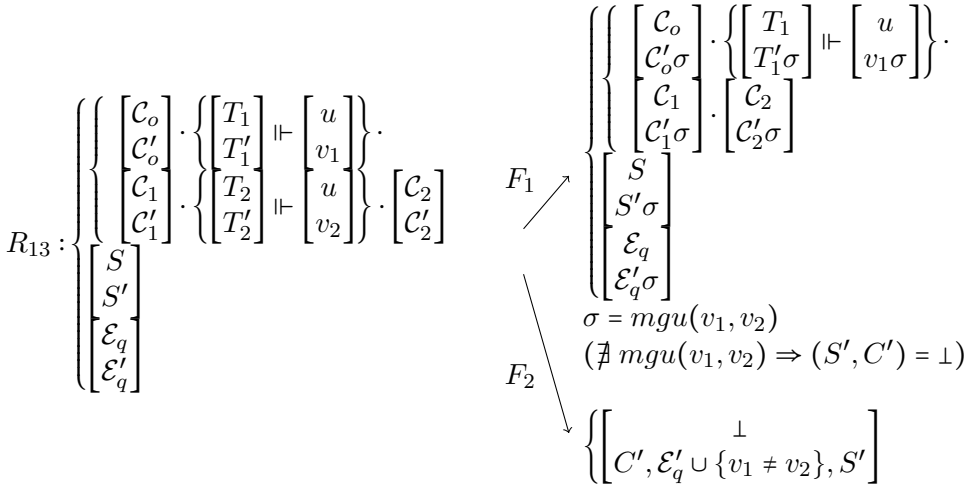


FIGURE 12 – La règle R_{13}

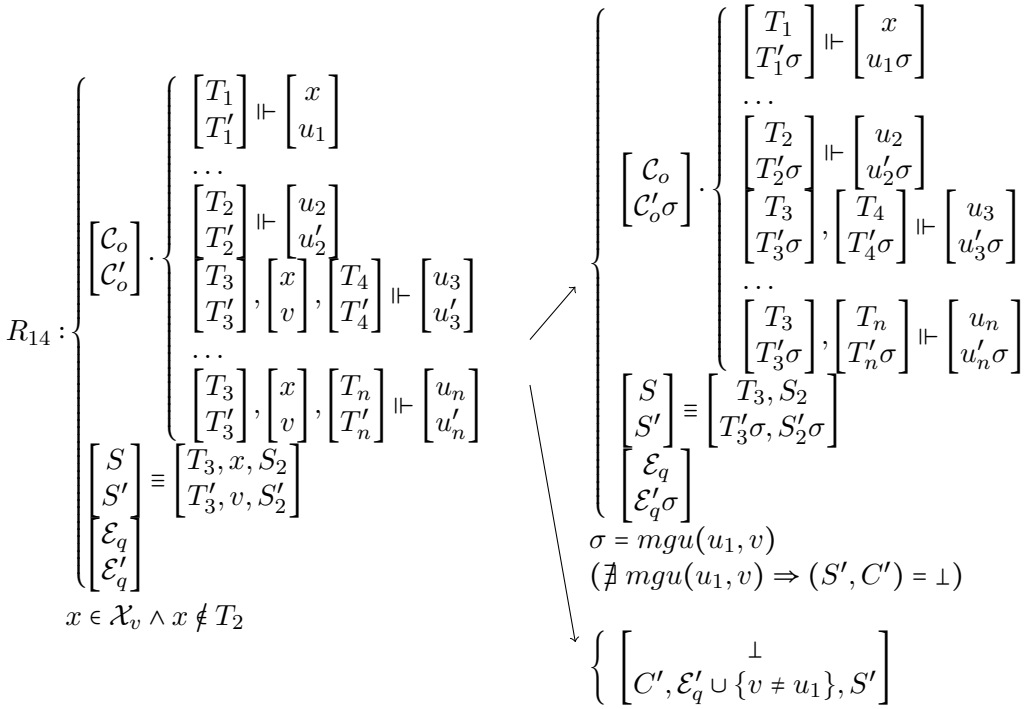


FIGURE 13 – La règle R_{14}

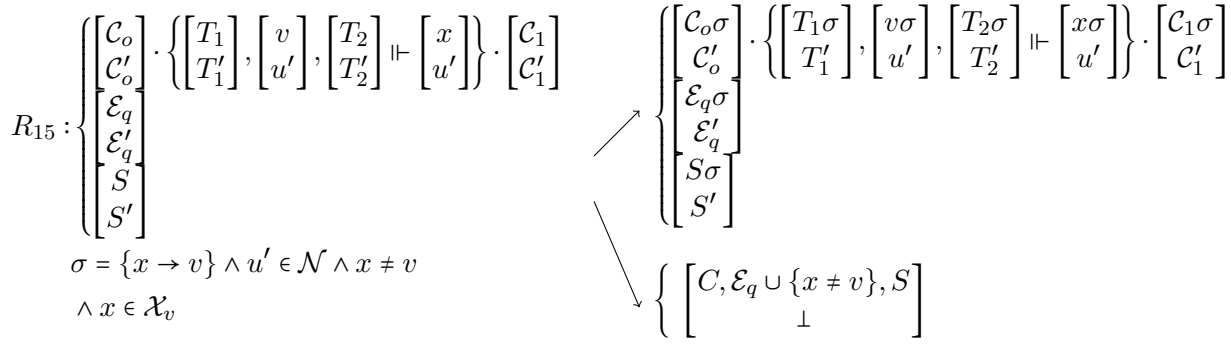
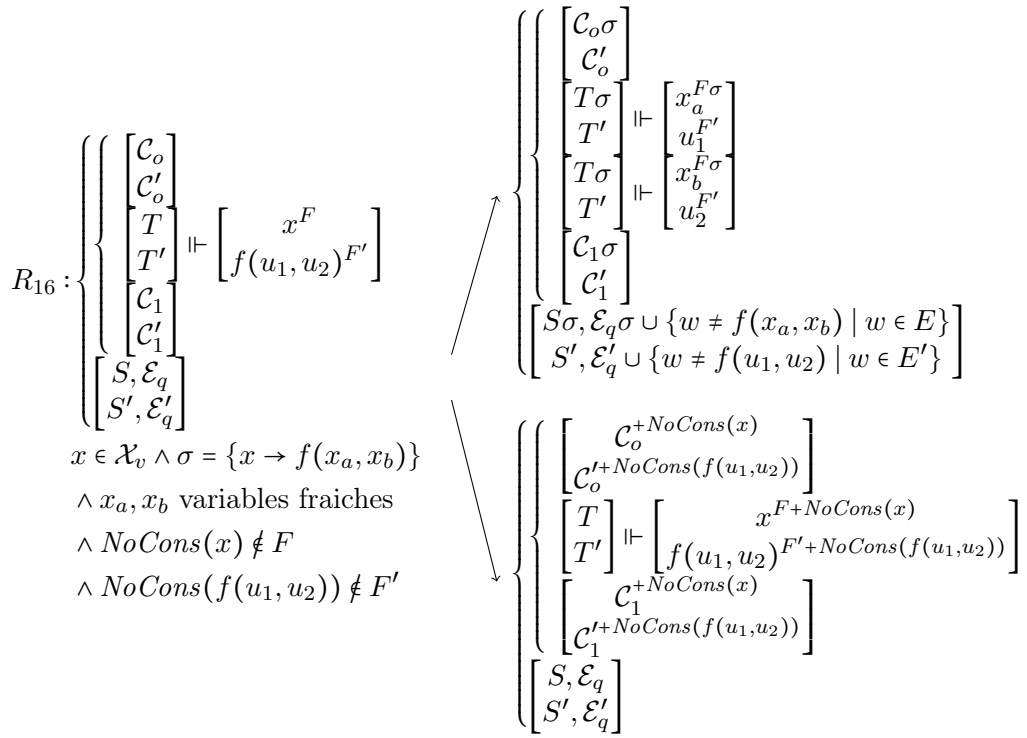
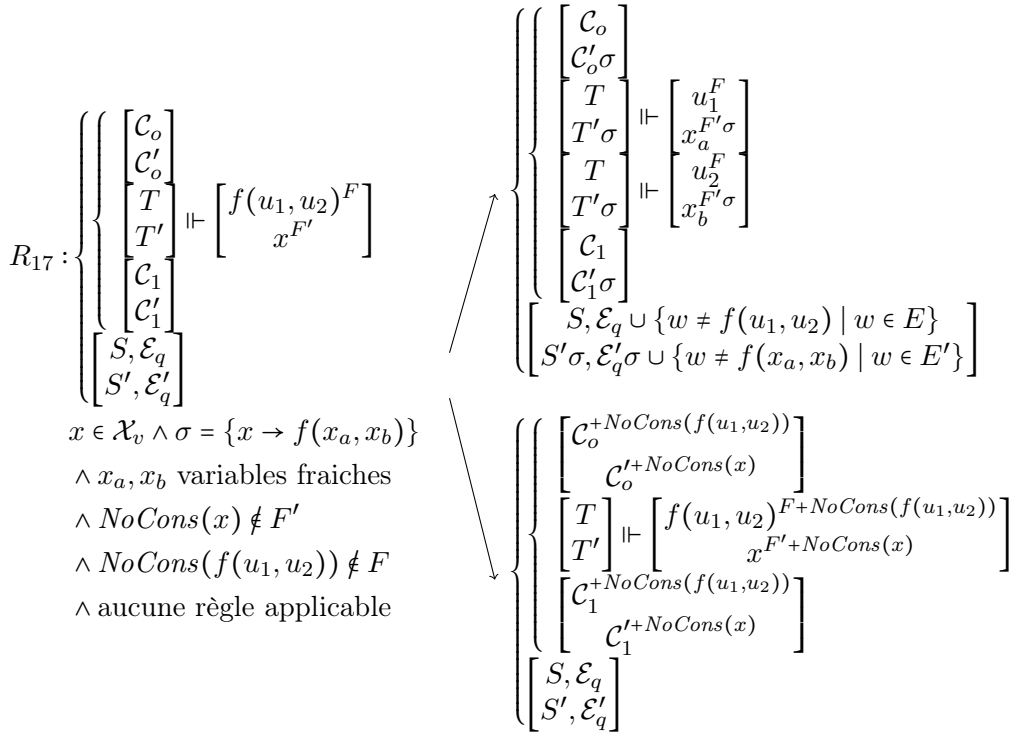


FIGURE 14 – La règle R_{15}



avec $E = \{w \mid NoCons(w) \in F\sigma \wedge \exists mgu(f(x_a, x_b), w)\}$
et $E' = \{w \mid NoCons(w) \in F' \wedge \exists mgu(f(u_1, u_2), w)\}$

FIGURE 15 – La règle R_{16}



avec $E = \{w \mid NoCons(w) \in F \wedge \exists mgu(f(u_1,u_2), w)\}$
et $E' = \{w \mid NoCons(w) \in F\sigma \wedge \exists mgu(f(x_a,x_b), w)\}$

FIGURE 16 – La règle R_{17}

B Exemples

Exemple B.1. Soit $S = [a, enc(x,b), y, << d, c >, b >]$, voici quelques exemples de preuves et de leurs applications sur S.

$$\begin{array}{lcl} \xi_1 = \pi_2(\pi_1(ax_4)) & \Rightarrow & \xi_1[S] = \pi_2(\pi_1(\langle\langle d,c\rangle,b\rangle)) \\ \xi_2 = dec(ax_2, \pi_2(ax_4)) & \Rightarrow & \xi_2[S] = dec(enc(x,b), \pi_2(\langle\langle d,c\rangle,b\rangle)) \\ \xi_3 = dec(ax_2, ax_3) & \Rightarrow & \xi_3[S] = dec(enc(x,b), y) \end{array}$$

Après normalisation, on obtient les termes suivants :

$$\begin{array}{l} \xi_1[S]\downarrow = c \\ \xi_2[S]\downarrow = x \\ \xi_3[S]\downarrow = dec(enc(x,b), y) \end{array}$$

Exemple B.2. Donnons un exemple de contrainte :

$$a, b, \langle enc(a,c), b\rangle, enc(d,c) \Vdash enc(y,c)$$

Dans cette contrainte, l'attaquant a la connaissance de 4 messages : a, b, $\langle enc(a,c), b\rangle$, $enc(d,c)$ et il doit réussir à construire le message $enc(y,c)$ (où y est une variable) à partir de sa connaissance en appliquant une recette (def 2.1) à son ensemble de connaissance. Dans ce cas, l'attaquant peut utiliser les recettes $\xi_1 = \pi_1(ax_3)$ ou $\xi_2 = ax_4$. Avec ξ_1, la variable y sera instanciée par a tandis qu'avec ξ_2, la variable y sera instanciée par d.

Exemple B.3. On reprend l'exemple du système de contraintes C défini dans l'exemple 2.1 et la solution définie dans l'exemple 2.2. Comme C est sans annotation, (σ, ξ_1, ξ_2) est bien solution du système de contraintes. Par contre, si on prend maintenant le même système de contraintes en rajoutant une annotation :

$$\begin{array}{lcll} a, b & \Vdash & x & \\ a, b, \langle enc(x,c), b\rangle & \Vdash & enc(y,c)^{NoPi(\langle enc(a,c),b\rangle)} & \mathcal{E}_q = \{z = \langle x, a\rangle\} \end{array}$$

alors $(\sigma, \langle ax_1, ax_1\rangle, \pi_1(ax_3))$ n'est plus solution du système de contraintes puisque $ax_3[T_2\sigma] = \langle enc(a,c), b\rangle$ ce qui est interdit par l'annotation $NoPi(u)$ avec le terme $u = \langle enc(a,c), b\rangle$.

Exemple B.4. Le prédicat des solutions en forme normale est donné par : $\mathcal{P}_N : \forall i \in [1..n], \xi_i\downarrow = \xi_i$. Ainsi toute les solutions de $\mathcal{S}_{\mathcal{P}_N}(C)$ seront bien en forme normale. Si on reprend le système de contraintes de l'exemple 2.1 et soit (σ, ξ_1, ξ_2) la solution donnée dans l'exemple 2.2, alors $(\sigma, \xi_1, \xi_2) \in \mathcal{S}_{\mathcal{P}_N}(C)$.

Voici trois exemples de prédicats qui correspondent à des supports complets pour l'équivalence symbolique. (On ne le démontrera pas car ce ne sont pas ces supports que nous choisirons au final, néanmoins les preuves sont faciles à retrouver).

Définition B.1 (Prédicats $\mathcal{P}_N$, $\mathcal{P}_{enc}$, $\mathcal{P}_{bouc}$)**.** Soit C un système de contrainte. On définit les trois prédicats $\mathcal{P}_N, \mathcal{P}_{enc}$ et $\mathcal{P}_{bouc}$: $\forall(\sigma, \xi_1, ..., \xi_n) \in Sol(C)$,

$$\begin{aligned}
\mathcal{P}_N : &\quad \forall i \in [1..n], \xi_i\downarrow = \xi_i \\
\mathcal{P}_{enc} : &\quad \forall i \in [1 \dots n], \nexists\, (\xi_a, \xi_b, \xi_c) \in \Pi^3 \text{ tq } \xi_i \in St(dec(enc(\xi_a, \xi_b), \xi_c))) \\
\mathcal{P}_{bouc} : &\quad \forall i \in [1 \dots n], \nexists\, (\xi_a, \xi_b) \in \Pi^2 \text{ tq } \xi_a \in St(\xi_i) \wedge \xi_b \in St(\xi_a) \wedge \xi_a \neq \xi_b \\
&\qquad \wedge \xi_a[T_i\sigma]\downarrow = \xi_b[T_i\sigma]\downarrow
\end{aligned}$$

Exemple B.5. Soit C le système de contraintes suivant :

$$\begin{array}{lcl} a, b & \Vdash & x \\ a, b, \langle enc(x, x), b\rangle & \Vdash & y \end{array} \qquad \mathcal{E}_q = \{z = \langle x, a\rangle\}$$

Soient (σ, ξ_1, ξ_2) et $(\sigma', \xi_1', \xi_2')$ deux triplets définis comme suit :

$$\begin{aligned}
\sigma &= \{x \to a; y \to a; z \to \langle a, a\rangle\} \\
\xi_1 &= ax_1 \\
\xi_2 &= dec(\pi_1(ax_3), dec(\pi_1(ax_3), ax_1))
\end{aligned}$$

$$\begin{aligned}
\sigma' &= \{x \to \langle a, a\rangle; y \to b; z \to \langle\langle a, a\rangle, a\rangle\} \\
\xi_1' &= \langle ax_1, ax_1\rangle \\
\xi_2' &= dec(enc(\pi_2(ax_3), ax_2), \pi_2(ax_3))
\end{aligned}$$

On a $(\sigma, \xi_1, \xi_2) \in Sol(C)$, $(\sigma', \xi_1', \xi_2') \in Sol(C)$ et :

$$\begin{gathered}
(\sigma, \xi_1, \xi_2) \in \mathcal{S}_{\mathcal{P}_N}(C), (\sigma, \xi_1, \xi_2) \in \mathcal{S}_{\mathcal{P}_{enc}}(C), (\sigma, \xi_1, \xi_2) \notin \mathcal{S}_{\mathcal{P}_{bouc}}(C) \\
(\sigma', \xi_1', \xi_2') \in \mathcal{S}_{\mathcal{P}_N}(C), (\sigma', \xi_1', \xi_2') \notin \mathcal{S}_{\mathcal{P}_{enc}}(C), (\sigma', \xi_1', \xi_2') \notin \mathcal{S}_{\mathcal{P}_{bouc}}(C)
\end{gathered}$$

C Preuves des théorèmes

C.1 Complétude de $\mathcal{S}_{\mathcal{P}_{un}}(C)$

Lemme 2.1 (Unicité des solutions du premier ordre)**.** *Soit C un système de contraintes.*

$$\forall \sigma, \sigma', \forall (\xi_1, ..., \xi_n) \in \Pi^n, \begin{cases} (\sigma, \xi_1, ..., \xi_n) \in Sol(C) \\ (\sigma', \xi_1, ..., \xi_n) \in Sol(C) \end{cases} \Rightarrow \sigma = \sigma'$$

Preuve du lemme 2.1. Soit $C = (\mathcal{C}_o, \mathcal{E}_q)$ un système de contraintes avec $\mathcal{C}_o = [T_1 \Vdash u_1; \ldots; T_n \Vdash u_n]$ et soient $(\sigma, \xi_1, \ldots, \xi_n)$ et $(\sigma', \xi_1, \ldots, \xi_n)$ deux solutions de C. On va démontrer par récurrence sur le nombre n de contraintes que $\forall x \in Var(\mathcal{C}_o), x\sigma = x\sigma'$.

Pour $n = 1$: Par définition d'un système de contraintes, T_1 est clos d'où :

$$\begin{aligned} \xi_1[T_1\sigma]\downarrow = \xi_1[T_1]\downarrow = \xi_1[T_1\sigma']\downarrow &\Rightarrow \xi_1[T_1\sigma]\downarrow = \xi_1[T_1\sigma']\downarrow \\ &\Rightarrow u_1\sigma = u_1\sigma' \\ &\Rightarrow \forall x \in Var(u_1), x\sigma = x\sigma' \\ &\Rightarrow \forall x \in Var(\mathcal{C}_o), x\sigma = x\sigma' \end{aligned}$$

Pour $n > 1$: Soit $\mathcal{C}'_o = [T_1 \Vdash u_1; \ldots; T_{n-1} \Vdash u_{n-1}]$. Par hypothèse de récurrence, $\forall x \in \mathcal{C}'_o, x\sigma = x\sigma'$ d'où $\forall x \in Var(T_n), x\sigma = x\sigma'$ (du fait qu'une variable ne puisse apparaître dans les membres gauches d'une contrainte que si elle a été déclarée dans un membre droit d'une contrainte précédente). D'où :

$$\begin{aligned} \forall x \in Var(T_n), x\sigma = x\sigma' &\Rightarrow T_n\sigma = T_n\sigma' \\ &\Rightarrow \xi_n[T_n\sigma]\downarrow = \xi_n[T_n\sigma']\downarrow \\ &\Rightarrow u_n\sigma = u_n\sigma' \\ &\Rightarrow \forall x \in Var(u_n), x\sigma = x\sigma' \\ &\Rightarrow \forall x \in Var(T_n \Vdash u_n), x\sigma = x\sigma' \\ &\Rightarrow \forall x \in Var(\mathcal{C}'_o) \cup Var(T_n \Vdash u_n), x\sigma = x\sigma' \\ &\quad \text{par l'hypothèse de récurrence} \\ &\Rightarrow \forall x \in Var(\mathcal{C}_o), x\sigma = x\sigma' \end{aligned}$$

On a bien $\forall x \in Var(\mathcal{C}_o), x\sigma = x\sigma'$. Or d'après la définition des systèmes de contraintes (définition 2.4), $\exists \alpha = mgu(\{u = v \mid (u = v) \in \mathcal{E}_q\})$ tq $Dom(\alpha) \cap$

$Var(\mathcal{C}_o) = \varnothing \wedge CoDom(\alpha) \subseteq Var(\mathcal{C}_o) \wedge Var(\mathcal{E}_q) \subseteq Var(\alpha) \cup Var(\mathcal{C}_o)$ donc comme σ et σ' sont des solutions du premier ordre, elles vérifient toutes les deux $\mathcal{E}_q$ d'où :

$$\begin{cases} \alpha = mgu(\{u = v \mid (u = v) \in \mathcal{E}_q\}) \\ \sigma, \sigma' \text{ solutions du premier ordre} \end{cases} \Rightarrow \exists \theta, \theta' \text{ tq } \sigma = \alpha\theta \wedge \sigma' = \alpha\theta'$$

De plus, comme α est un mgu, alors $Dom(\alpha) \cap CoDom(\alpha)$ d'où $\forall x \in Dom(\alpha), x\alpha\alpha = x\alpha$. On a donc :

$$\begin{cases} CoDom(\alpha) \subseteq Var(\mathcal{C}_o) \\ \forall x \in Var(\mathcal{C}_o), x\sigma = x\sigma' \end{cases} \Rightarrow \forall x \in CoDom(\alpha), x\sigma = x\sigma'$$

$$\Rightarrow \forall y \in Dom(\alpha), y\alpha\sigma = y\alpha\sigma'$$

car $\forall y \in Dom(\alpha), Var(y\alpha) \subseteq CoDom(\alpha)$

$$\Rightarrow \forall y \in Dom(\alpha), y\alpha\alpha\theta = y\alpha\alpha\theta'$$

car $\forall y \in Dom(\alpha), y\alpha\alpha = y\alpha$

$$\Rightarrow \forall y \in Dom(\alpha), y\alpha\theta = y\alpha\theta'$$

$$\Rightarrow \forall y \in Dom(\alpha), y\sigma = y\sigma'$$

On a donc $\forall y \in Dom(\alpha), y\sigma = y\sigma'$ et $\forall x \in CoDom(\alpha), x\sigma = x\sigma'$ d'où $\forall x \in Var(\alpha), x\sigma = x\sigma'$. Comme $Var(\mathcal{E}_q) \subseteq Var(\alpha) \cup Var(\mathcal{C}_o)$, on a donc $\forall x \in \mathcal{E}_q, x\sigma = x\sigma'$.

Conclusion : $\sigma = \sigma'$.

□

Pour faciliter l'écriture des preuves, nous allons introduire un nouvel outil sur les termes : *les contextes de preuves.*

Définition C.1 (Contexte de preuve)**.** Un *contexte de preuve*, noté $\xi\lfloor\_\rfloor$, est un élément de $\Pi_c = \mathcal{T}(\mathcal{C} \cup \mathcal{D} \cup \{\_\}, \mathcal{AX})$. Soit $(\xi, \xi') \in \Pi_c^2$, on définit $\xi\lfloor\xi'\rfloor$ par récurrence sur ξ :

- $\forall x \in \mathcal{X}, x\lfloor\xi'\rfloor = x$
- $\forall c \in \mathcal{N}, c\lfloor\xi'\rfloor = c$
- $\_\lfloor\xi'\rfloor = \xi'$
- $\forall f \in \mathcal{C} \cup \mathcal{D}, f(\xi_1, ..., \xi_m)\lfloor\xi'\rfloor = f(\xi_1\lfloor\xi'\rfloor, ..., \xi_m\lfloor\xi'\rfloor)$

Exemple C.1. Exemple de preuves et contextes de preuve :

$$\begin{aligned}
\xi_1\lfloor_\rfloor &= \pi_1(\pi_2(_)) \\
\xi_2\lfloor_\rfloor &= dec(\langle_, b\rangle, _) \\
\xi_3 &= \pi_2(ax_5) \\
\xi_1\lfloor\xi_2\rfloor &= \pi_1(\pi_2(dec(\langle_, b\rangle, _))) \\
\xi_2\lfloor\xi_1\rfloor &= dec(\langle\pi_1(\pi_2(_)), b\rangle, \pi_1(\pi_2(_))) \\
\xi_2\lfloor\xi_1\rfloor\lfloor\xi_3\rfloor &= dec(\langle\pi_1(\pi_2(\pi_2(ax_5))), b\rangle, \pi_1(\pi_2(\pi_2(ax_5))))
\end{aligned}$$

En utilisant le formalisme ci-dessus, on redonne la définition des supports $\mathcal{S}_{\mathcal{P}_{bouc}}(C), \mathcal{S}_{\mathcal{P}_N}(C), \mathcal{S}_{\mathcal{P}_{enc}}(C)$:

Définition C.2 (Prédicats $\mathcal{P}_N$, $\mathcal{P}_{enc}$, $\mathcal{P}_{bouc}$)**.** Soit C un système de contrainte. On définit les trois prédicats $\mathcal{P}_N, \mathcal{P}_{enc}$ et $\mathcal{P}_{bouc}$: $\forall(\sigma, \xi_1, ..., \xi_n) \in Sol(C)$,

$$\begin{aligned}
\mathcal{P}_N : &\quad \forall i \in [1..n], \xi_i\downarrow = \xi_i \\
\mathcal{P}_{enc} : &\quad \forall i, \nexists\, \xi_a, \xi_b, \xi_c,\ \xi_d \text{ tq } \xi_i = \xi_a\lfloor dec(enc(\xi_b, \xi_c), \xi_d))\rfloor \\
\mathcal{P}_{bouc} : &\quad \forall i, \nexists\, \xi_a, \xi_b, \xi_c \text{ tq } \xi_i = \xi_a\lfloor\xi_b\lfloor\xi_c\rfloor\rfloor \wedge \xi_b \neq _ \wedge (\xi_b\lfloor\xi_c\rfloor)[T_i\sigma]\downarrow = \xi_c[T_i\sigma]\downarrow
\end{aligned}$$

Lemme 3.1 (Règles de récriture et $\mathcal{S}_{\mathcal{P}_{bouc}}$)**.** *Soit $C = (\mathcal{C}_o, \mathcal{E}_q)$ un système de contraintes avec $\mathcal{C}_o = [T_1 \Vdash u_1; \ldots; T_n \Vdash u_n]$.* $\forall(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{bouc}}(C), \forall i \in [1..n], \forall \xi_a \in \Pi_c, \forall \xi_b \in \Pi$,

$$\xi_i = \xi_a\lfloor\xi_b\rfloor \Rightarrow \xi_b[T_i\sigma]\downarrow \in \mathcal{T}(\mathcal{C} \cup \mathcal{N})$$

Preuve du lemme 3.1. Soit C un système de contraintes. Soit $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{bouc}}(C)$. Soit $i \in [1..n]$.

Supposons $\exists \xi_a, \xi_b$ tq $\xi_i = \xi_a\lfloor\xi_b\rfloor \wedge \xi_b[T_i\sigma]\downarrow \notin \mathcal{T}(\mathcal{C} \cup \mathcal{N})$. Comme $(\sigma, \xi_1, ..., \xi_n) \in Sol(C)$, alors $\xi_i[T_i\sigma]\downarrow \in \mathcal{T}(\mathcal{C} \cup \mathcal{N})$ et donc $\xi_a\lfloor\xi_b\rfloor[T_i\sigma]\downarrow \in \mathcal{T}(\mathcal{C} \cup \mathcal{N})$ ce qui implique que $\xi_b[T_i\sigma]\downarrow$ a été réduit par une des règles de récriture. Donc d'après les règles de récriture, $\xi_i[T_i\sigma]\downarrow \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}) \Rightarrow \exists \xi_c \in \Pi_c, (\xi_d, \xi_e, \xi_f) \in \Pi^3$ tq :

$$\begin{gathered}
\xi_i[T_i\sigma]\downarrow = \xi_c\lfloor dec(enc(\xi_d[T_i\sigma]\downarrow, u), u)\rfloor[T_i\sigma]\downarrow \\
\text{ou} \\
\xi_i[T_i\sigma]\downarrow = \xi_c\lfloor\pi_1(\langle\xi_d[T_i\sigma]\downarrow, u\rangle)\rfloor[T_i\sigma]\downarrow \\
\text{ou} \\
\xi_i[T_i\sigma]\downarrow = \xi_c\lfloor\pi_2(\langle u, \xi_d[T_i\sigma]\downarrow\rangle)\rfloor[T_i\sigma]\downarrow
\end{gathered}$$

avec

$$\xi_i = \xi_c\lfloor dec(enc(\xi_d, \xi_e), \xi_f)\rfloor \wedge \xi_b \in St(\xi_e) \cup St(\xi_f)$$
ou
$$\xi_i = \xi_c\lfloor \pi_1(\langle \xi_d, \xi_e\rangle)\rfloor \wedge \xi_b \in St(\xi_e)$$
ou
$$\xi_i = \xi_c\lfloor \pi_2(\langle \xi_e, \xi_d\rangle)\rfloor \wedge \xi_b \in St(\xi_e)$$

d'où :

$$dec(enc(\xi_d[T_i\sigma]\downarrow, u), u)\downarrow = \xi_d[T_i\sigma]\downarrow$$
ou
$$\pi_1(\langle \xi_d[T_i\sigma]\downarrow, u\rangle)\downarrow = \xi_d[T_i\sigma]\downarrow$$
ou
$$\pi_2(\langle u, \xi_d[T_i\sigma]\downarrow\rangle)\downarrow = \xi_d[T_i\sigma]\downarrow$$

d'où

$$\exists \xi_B \in \Pi_c \text{ tq } \xi_i = \xi_c\lfloor \xi_B\rfloor\lfloor \xi_d\rfloor \wedge \xi_B\lfloor \xi_d\rfloor[T_i\sigma]\downarrow = \xi_d[T_i\sigma]\downarrow$$

avec

$$\xi_B\lfloor \_\rfloor = dec(enc(\_, \xi_e), \xi_f)$$
ou
$$\xi_B\lfloor \_\rfloor = \pi_1(\langle \_, \xi_e\rangle)$$
ou
$$\xi_B\lfloor \_\rfloor = \pi_2(\langle \xi_e, \_\rangle)$$

ce qui est en contradiction avec le fait que $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{bouc}}(C)$. □

Lemme 3.2 (Destructeur dans $\mathcal{S}_{\mathcal{P}_{bouc}}$)**.** *Soit* $C = (\mathcal{C}_o, \mathcal{E}_q)$ *un système de contraintes avec* $\mathcal{C}_o = [T_1 \Vdash u_1; \ldots; T_n \Vdash u_n]$. $\forall (\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{bouc}}, \forall i \in [1..n], \xi_i \notin \mathcal{T}(\mathcal{C}, \mathcal{AX}) \Rightarrow \exists p \in [1 \ldots |T_i|], \xi \in \Pi_c, \xi' \in \Pi$ tq

$$\xi_i = \xi\lfloor dec(ax_p, \xi')\rfloor \vee \xi_i = \xi\lfloor \pi_1(ax_p)\rfloor \vee \xi_i = \xi\lfloor \pi_2(ax_p)\rfloor$$

Preuve du lemme 3.2. Soit $C = (\mathcal{C}_o, \mathcal{E}_q)$ un système de contraintes avec $\mathcal{C}_o = [T_1 \Vdash u_1; \ldots; T_n \Vdash u_n]$. Soient $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{bouc}}(C), i \in [1..n]$. Effectuons une preuve par récurrence sur la taille de ξ_i :

Si $|\xi_i| = 1$: Dans ce cas, $\exists p$ tq $\xi_i = ax_p$. ξ_i ne peut contenir de destructeur d'où $\xi_i \in \mathcal{T}(\mathcal{C}, \mathcal{AX})$ et donc le lemme est vériié.

Si $|\xi_i| > 1$: Dans ce cas, soit $\xi_i \in \mathcal{T}(\mathcal{C}, \mathcal{AX})$ et donc le lemme est vérifié, soit $\xi_i \notin \mathcal{T}(\mathcal{C}, \mathcal{AX})$ et donc on a différent cas possibles :

- $\xi_i = enc(\xi_a, \xi_b)$
- $\xi_i = \langle \xi_a, \xi_b\rangle$

– $\xi_i = dec(\xi_a, \xi_b)$
– $\xi_i = \pi_1(\xi_a)$
– $\xi_i = \pi_2(\xi_a)$

Les cas où $Top(\xi_i) \in \mathcal{C}$ sont triviaux. En effet, on sait que $\xi_i \notin \mathcal{T}(\mathcal{C}, \mathcal{AX})$ et $Top(\xi_i) \in \mathcal{C}$, d'où $\xi_a \notin \mathcal{T}(\mathcal{C}, \mathcal{AX})$ ou $\xi_b \notin \mathcal{T}(\mathcal{C}, \mathcal{AX})$. Comme, $|\xi_a| < |\xi_i|$ et $|\xi_b| < |\xi_i|$, on peut appliquer l'hypothèse de récurrence sur ξ_a et ξ_b ce qui vérifie le lemme. Voyons le cas $\xi_i = dec(\xi_a, \xi_b)$. Deux possibilités s'offrent à nous :

1. $\xi_a \notin \mathcal{T}(\mathcal{C}, \mathcal{AX})$ ou $\xi_b \notin \mathcal{T}(\mathcal{C}, \mathcal{AX})$
2. $\xi_a \in \mathcal{T}(\mathcal{C}, \mathcal{AX})$ et $\xi_b \in \mathcal{T}(\mathcal{C}, \mathcal{AX})$

La première possibilité est similaire au cas où $Top(\xi_i) \in \mathcal{C}$: On applique l'hypothèse de récurrence du ξ_a et ξ_b ce qui permet de montrer le lemme. Il reste donc la deuxième possibilité : $\xi_a \in \mathcal{T}(\mathcal{C}, \mathcal{AX})$ et $\xi_b \in \mathcal{T}(\mathcal{C}, \mathcal{AX})$. Pour vérifié le lemme, il faut donc montrer qu'il existe $p \in [1 \dots |T_i|]$ tq $\xi_a = ax_p$. Supposons que ce p n'existe pas : On sait que $(\sigma, \xi_1, \dots, \xi_n) \in Sol(C)$ donc $\xi_i[T_i\sigma]\downarrow \in \mathcal{T}(\mathcal{C} \cup \mathcal{N})$. Prenons le cas où $\xi_i = dec(\xi_a, \xi_b)$:

$$
\begin{aligned}
\xi_i[T_i\sigma]\downarrow \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}) &\Rightarrow dec(\xi_a[T_i\sigma]\downarrow, \xi_b[T_i\sigma]\downarrow)\downarrow \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}) \\
&\Rightarrow \exists (u_a, u_b) \in \mathcal{T}(\mathcal{C} \cup \mathcal{N})^2 \text{ tq } \xi_a[T_i\sigma]\downarrow = enc(u_a, u_b) \wedge \\
&\quad \xi_b[T_i\sigma]\downarrow = u_b \wedge \xi_i[T_i\sigma]\downarrow = u_a \\
&\Rightarrow \exists (\xi_c, \xi_d) \in \Pi_c \text{ tq } \xi_a = enc(\xi_c, \xi_d) \wedge \xi_c[T_i\sigma]\downarrow = u_a \wedge \\
&\quad \xi_d[T_i\sigma]\downarrow = u_d \wedge \xi_i[T_i\sigma]\downarrow = u_a \\
&\qquad \text{car } \xi_a \in \mathcal{T}(\mathcal{C}, \mathcal{AX}) \text{ et } \forall p \in [1 \dots |T_i|], \xi_a \neq ax_p \\
&\Rightarrow \xi_i[T_i\sigma]\downarrow = \xi_c[T_i\sigma]\downarrow \wedge \xi_i = dec(enc(\xi_c, \xi_d), \xi_b) \\
&\Rightarrow (\sigma, \xi_1, \dots, \xi_n) \notin \mathcal{S}_{\mathcal{P}_{bouc}}(C)
\end{aligned}
$$

On a donc une absurdité ce qui permet de conclure que $\exists p \in [1 \dots |T_i|]$ tq $\xi_a = ax_p$ d'où $\xi_i = dec(ax_p, \xi_b)$. On effectue une preuve similaire pour le cas $\xi_i = \pi_1(\xi_a)$ et $\xi_i = \pi_2(\xi_b)$ ce qui permet de conclure la validité du lemme. □

Lemme 3.3 (Inclusion de support)**.** *Soit C un système de contraintes :*
– $\mathcal{S}_{\mathcal{P}_{bouc}}(C) \subseteq \mathcal{S}_{\mathcal{P}_N}(C)$
– $\mathcal{S}_{\mathcal{P}_{bouc}}(C) \subseteq \mathcal{S}_{\mathcal{P}_{enc}}(C)$

Preuve du lemme 3.3. Soit C un système de contraintes.
$\underline{\mathcal{S}_{\mathcal{P}_{bouc}}(C) \subseteq \mathcal{S}_{\mathcal{P}_N}(C)}$:

Soit $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{bouc}}(C)$. On raisonne par l'absurde :

$$
\begin{aligned}
(\sigma, \xi_1, ..., \xi_n) \notin \mathcal{S}_{\mathcal{P}_N}(C) &\Rightarrow \exists i \text{ tq } \xi_i\downarrow \neq \xi_i \\
&\Rightarrow \exists i, \xi_a, \xi_b, \xi_c \text{ tq } \begin{cases} \xi_i = \xi_a\lfloor \pi_1(\langle \xi_b, \xi_c \rangle)\rfloor \\ ou \\ \xi_i = \xi_a\lfloor \pi_2(\langle \xi_c, \xi_b \rangle)\rfloor \\ ou \\ \xi_i = \xi_a\lfloor dec(enc(\xi_b, \xi_c), \xi_c)\rfloor \end{cases} \\
&\Rightarrow \begin{cases} \pi_1(< \xi_b, \xi_c >)[T_i\sigma]\downarrow = \xi_b[T_i\sigma]\downarrow \\ ou \\ \pi_2(< \xi_c, \xi_b >)[T_i\sigma]\downarrow = \xi_b[T_i\sigma]\downarrow \\ ou \\ dec(enc(\xi_b, \xi_c), \xi_c)[T_i\sigma]\downarrow = \xi_b[T_i\sigma]\downarrow \end{cases} \\
&\Rightarrow \exists \xi_C \in \Pi_c \text{ tq } \xi_i = \xi_a\lfloor \xi_C\lfloor \xi_b \rfloor\rfloor \wedge \xi_C\lfloor \xi_b \rfloor[T_i\sigma]\downarrow = \xi_b[T_i\sigma]\downarrow \\
&\Rightarrow (\sigma, \xi_1, ..., \xi_n) \notin \mathcal{S}_{\mathcal{P}_{bouc}}(C)
\end{aligned}
$$

On aboutit bien à une absurdité d'où $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_N}(C)$

$\underline{\mathcal{S}_{\mathcal{P}_{bouc}}(C) \subseteq \mathcal{S}_{\mathcal{P}_{enc}}(C)}$:

Soit $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{bouc}}(C)$. On raisonne par l'absurde :

$$
\begin{aligned}
(\sigma, \xi_1, ..., \xi_n) \notin \mathcal{S}_{\mathcal{P}_{enc}}(C) &\Rightarrow \exists i, \xi_a, \xi_b, \xi_c, \xi_d \text{ tq } \xi_i = \xi_a\lfloor dec(enc(\xi_b, \xi_c), \xi_d)\rfloor \\
&\Rightarrow dec(enc(\xi_b, \xi_c), \xi_d)[T_i\sigma]\downarrow \in \mathcal{T}(\mathcal{C} \cup \mathcal{N}) \\
&\quad \text{par le lemme 3.1} \\
&\Rightarrow dec(enc(\xi_b, \xi_c), \xi_d)[T_i\sigma]\downarrow = \xi_b[T_i\sigma]\downarrow \\
&\Rightarrow \exists \xi_C \in \Pi_c \text{ tq } \xi_i = \xi_a\lfloor \xi_C\lfloor \xi_b \rfloor\rfloor \wedge \xi_C\lfloor \xi_b \rfloor[T_i\sigma]\downarrow = \xi_b[T_i\sigma]\downarrow \\
&\quad \text{avec } \xi_C = dec(enc(\_, \xi_c), \xi_d) \\
&\Rightarrow (\sigma, \xi_1, ..., \xi_n) \notin \mathcal{S}_{\mathcal{P}_{bouc}}(()C)
\end{aligned}
$$

On aboutit également à une absurdité d'où $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{enc}}(C)$ □

Lemme 3.4 (Inclusion de support)**.** *Soit C un système de contraintes,* $\mathcal{S}_{\mathcal{P}_{un}}(C) \subseteq \mathcal{S}_{\mathcal{P}_{bouc}}(C)$.

Preuve du lemme 3.4. Soit C un système de contraintes. Soit $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C)$. Supposons que $(\sigma, \xi_1, ..., \xi_n) \notin \mathcal{S}_{\mathcal{P}_{bouc}}(C)$.

$$
\begin{aligned}
(\sigma, \xi_1, ..., \xi_n) \notin \mathcal{S}_{\mathcal{P}_{bouc}}(C) \Rightarrow\ & \exists i \in [1 \dots n], \exists (\xi_a, \xi_b) \in \Pi_c^2, \xi_c \in \Pi \text{ tq } \xi_i = \xi_a \lfloor \xi_b \lfloor \xi_c \rfloor \rfloor \\
& \wedge\ \xi_b \lfloor \xi_c \rfloor [T_i\sigma]\downarrow = \xi_c [T_i\sigma]\downarrow \wedge \xi_b \neq \_ \\
& \Rightarrow \exists i \in [1 \dots n], (\xi_A, \xi_C) \in \Pi_c^2, (\xi_B, \xi_D) \in \Pi^2 \text{ tq} \\
& \xi_i = \xi_A \lfloor \xi_B \rfloor = \xi_C \lfloor \xi_D \rfloor \wedge \xi_B [T_i\sigma]\downarrow = \xi_D [T_i\sigma]\downarrow \\
& \quad \text{avec } \xi_A = \xi_a, \xi_B = \xi_b \lfloor \xi_c \rfloor, \xi_C = \xi_a \lfloor \xi_b \rfloor \text{ et } \xi_D = \xi_c \\
& \Rightarrow (\sigma, \xi_1, ..., \xi_n) \notin \mathcal{S}_{\mathcal{P}_{un}}(C) \\
& \quad \text{car } \xi_B \neq \xi_D
\end{aligned}
$$

On aboutit à une absurdité d'où $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{bouc}}(C)$. □

Lemme 3.5 (Relation entre $\mathcal{S}_{\mathcal{P}_{un}}$ et l'équivalence statique)**.** *Soit C et C' deux systèmes de contraintes et S, S' deux séquences de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$. Si $(\sigma, \xi_1, ..., \xi_n) \in Sol(C)$,*
$(\sigma', \xi_1, ..., \xi_n) \in Sol(C')$ et $S\sigma \sim S'\sigma'$, alors :

$$(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C) \Leftrightarrow (\sigma', \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C')$$

Preuve du lemme 3.5. Soient C et C' deux systèmes de contraintes. Soient S et S' deux séquences de termes. Soient $(\sigma, \xi_1, \dots, \xi_n) \in Sol(C)$ et $(\sigma', \xi_1, \dots, \xi_n) \in Sol(C')$ telles que $S\sigma \sim S'\sigma'$.

$\underline{(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C) \Rightarrow (\sigma', \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C')}$: On va démontrer que $(\sigma', \xi_1, ..., \xi_n)$ vérifie bien le prédicat $\mathcal{P}_{un}$ (définition 3.6). $\forall i, j \in [1..n]^2, \forall \xi_a, \xi_b, \xi_c, \xi_d$, (supposons que $i \leq j$, le cas $j \leq i$ se démontrera par symétrie entre i et j)

$$\begin{cases} i \le j \\ \xi_i = \xi_a\lfloor\xi_b\rfloor \\ \xi_j = \xi_c\lfloor\xi_d\rfloor \\ \xi_b[T'_i\sigma']\downarrow = \xi_d[T'_j\sigma']\downarrow \end{cases} \Rightarrow \begin{cases} i \le j \\ \xi_i = \xi_a\lfloor\xi_b\rfloor \\ \xi_j = \xi_c\lfloor\xi_d\rfloor \\ \xi_b[T'_j\sigma']\downarrow = \xi_d[T'_j\sigma']\downarrow \end{cases}$$

$$\text{car } T_i \sqsubseteq T_j$$

$$\Rightarrow \begin{cases} i \le j \\ \xi_i = \xi_a\lfloor\xi_b\rfloor \\ \xi_j = \xi_c\lfloor\xi_d\rfloor \\ \xi_b[T_j\sigma]\downarrow = \xi_d[T_j\sigma]\downarrow \end{cases}$$

$$\text{car } S\sigma \sim S'\sigma \text{ d'où } T_j\sigma \sim T'_j\sigma'$$

$$\Rightarrow \begin{cases} i \le j \\ \xi_i = \xi_a\lfloor\xi_b\rfloor \\ \xi_j = \xi_c\lfloor\xi_d\rfloor \\ \xi_b[T_i\sigma]\downarrow = \xi_d[T_j\sigma]\downarrow \end{cases}$$

$$\text{car } |T'_i| = |T_i| \text{ et } \max\{j \mid ax_j \in \xi_b\} \le |T'_i|$$

$$\Rightarrow \xi_b = \xi_d$$

$$\text{car } (\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C)$$

$$\Rightarrow (\sigma', \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C')$$

D'où $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C) \Rightarrow (\sigma', \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C')$. La preuve de l'implication gauche est donnée par symétrie de cette preuve.

Conclusion : $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C) \Leftrightarrow (\sigma', \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C')$. □

Lemme 3.6 (De $Sol(C)$ à $\mathcal{S}_{\mathcal{P}_{un}}(C)$)**.** *Soit C un système de contraintes.* $\forall(\sigma, \xi_1, \ldots, \xi_n) \in Sol(C), \exists(\xi'_1, \ldots, \xi'_n) \in \Pi^n$ tq $(\sigma, \xi'_1, \ldots, \xi'_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C)$.

Preuve du lemme 3.6. Soit C un système de contraintes sans annotation et soit $(\sigma, \xi_1, \ldots, \xi_n) \in Sol(C)$. On veut démontrer que :

$$\exists(\xi'_1, \ldots, \xi'_n) \text{ tq } (\sigma, \xi'_1, \ldots, \xi'_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C)$$

Intuitivement, la démonstration va consister à donner un algorithme qui permet de supprimer les doublons de preuve du type $\xi_a[T\sigma]\downarrow = \xi_b[T\sigma]\downarrow$ avec $\xi_a \neq \xi_b$ en les remplaçant par une unique preuve.

Soit $(\sigma, \xi_1, \ldots, \xi_n) \in Sol(C)$. Pour un terme quelconque u, on définit l'ensemble suivant :

$$P_{\xi_1,...,\xi_n}(u) = \{\xi \in \Pi \mid \exists i \in [1..n] \text{ tq } \xi \in St(\xi_i) \wedge \xi[T_{max}(C)\sigma]\downarrow = u\}$$

$P_{\xi_1,\ldots,\xi_n}(u)$ est un ensemble de preuves disjointes (donc si la preuve ξ apparaît plusieurs fois dans $(\sigma, \xi_1, \ldots, \xi_n)$, elle n'apparaît en revanche qu'une seule fois dans $P_{\xi_1,\ldots,\xi_n}(u)$). On définit une mesure $\|P\|_{\xi_1,\ldots,\xi_n}$ par :

$$\|P\|_{\xi_1,\ldots,\xi_n} = \sum_{u \in \mathcal{T}(\mathcal{C} \cup \mathcal{D} \cup \mathcal{N})} \max(|P_{\xi_1,\ldots,\xi_n}(u)| - 1; 0)$$

où $|P_{\xi_1,\ldots,\xi_n}(u)|$ est le cardinal de l'ensemble $P_{\xi_1,\ldots,\xi_n}(u)$.

$\|P\|_{\xi_1,\ldots,\xi_n}$ est trivialement toujours positive et bornée (aucune preuve infinie). Montrons maintenant la propriété suivante :

$$\forall (\sigma, \xi_1, \ldots, \xi_n) \in Sol(C), \|P\|_{\xi_1,\ldots,\xi_n} = 0 \Rightarrow (\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C) \quad (1)$$

Supposons donc que $(\sigma, \xi_1, \ldots, \xi_n) \notin \mathcal{S}_{\mathcal{P}_{un}}(C)$:

$$\begin{aligned}
(\sigma, \xi_1, \ldots, \xi_n) \notin \mathcal{S}_{\mathcal{P}_{un}}(C) &\Rightarrow \exists (i,j) \in [1 \ldots n]^2, \exists \xi \in St(\xi_i), \exists \xi' \in St(\xi_j) \text{ tq} \\
&\quad \xi[T_i\sigma]\downarrow = \xi'[T_j\sigma]\downarrow \wedge \xi \neq \xi' \\
&\Rightarrow \exists u \in \mathcal{T}(\mathcal{C} \cup \mathcal{D} \cup \mathcal{N}) \text{ tq } \xi[T_i\sigma]\downarrow = \xi'[T_j\sigma]\downarrow = u \wedge \xi \neq \xi' \\
&\Rightarrow \xi \in P_{\xi_1,\ldots,\xi_n}(u) \wedge \xi' \in P_{\xi_1,\ldots,\xi_n}(u) \wedge \xi \neq \xi' \\
&\Rightarrow |P_{\xi_1,\ldots,\xi_n}(u)| \geq 2 \\
&\Rightarrow \max(|P_{\xi_1,\ldots,\xi_n}(u)| - 1; 0) \geq 1 \\
&\Rightarrow \|P\|_{\xi_1,\ldots,\xi_n} \neq 0
\end{aligned}$$

On a bien $\|P\|_{\xi_1,\ldots,\xi_n} = 0 \Rightarrow (\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C)$. C'est cette propriété qui nous permettra de conclure le lemme. On va en effet démontrer que $\forall (\sigma, \xi_1, \ldots, \xi_n) \in Sol(C), \exists (\xi'_1, \ldots, \xi'_n) \in \Pi^n$ tq $(\sigma, \xi'_1, \ldots, \xi'_n) \in Sol(C) \wedge \|P\|_{\xi'_1,\ldots,\xi'_n} = 0$. Et donc $(\sigma, \xi'_1, \ldots, \xi'_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C)$ d'après la propriété 1.

Soit $(\sigma, \xi_1, \ldots, \xi_n) \in Sol(C)$ et soit :

$$(\beta_1, \ldots, \beta_n) = \underset{(\xi'_1,\ldots,\xi'_n) \in Sol_2(C)}{\operatorname{argmin}} \{\|P\|_{\xi'_1,\ldots,\xi'_n} \mid \lambda_C(\xi'_1, \ldots, \xi'_n) = \sigma\}$$

Supposons que $\|P\|_{\beta_1,\ldots,\beta_n} > 0$ (sinon aucun intérêt). De ce fait, on sait qu'il existe $u \in \mathcal{T}(\mathcal{C} \cup \mathcal{D} \cup \mathcal{N})$ tq $P_{\beta_1,\ldots,\beta_n}(u) > 1$.

Notons $\min(\xi) = \min\{i \mid \xi \in St(\xi_i)\}$ et soit $\beta_0 \in P_{\beta_1,\ldots,\beta_n}(u)$ telle que :

$$\forall \xi' \in P_{\beta_1,\ldots,\beta_n}(u), \min(\beta_0) \leq \min(\xi') \wedge \forall \gamma \in St(\beta_0), \gamma[T_{max}(C)]\downarrow \neq u$$

Intuitivement, β_0 est une preuve de $P_{\beta_1,\ldots,\beta_n}(u)$ dont aucun de ses sous-termes ne démontre u et qui peut s'appliquer sur les mêmes contraintes que

toutes les autres preuves qui démontrent u. L'existence de β_0 est démontré par le fait que $P_{\beta_1,\ldots,\beta_n}(u)$ est un ensemble fini.

Soit F la fonction récursive définie sur Π par :

1. $F(ax_i) = ax_i$ si $ax_i[T_{max}(C)\sigma]\downarrow \neq u$
2. $F(ax_i) = \beta_0$ si $ax_i[T_{max}(C)\sigma]\downarrow = u$
3. $F(f(\alpha_1,\ldots,\alpha_n)) = f(F(\alpha_1),\ldots,F(\alpha_n))$ si $f(\alpha_1,\ldots,\alpha_n)[T_{max}(C)\sigma]\downarrow \neq u$
4. $F(f(\alpha_1,\ldots,\alpha_m)) = \beta_0$ si $f(\alpha_1,\ldots,\alpha_m)[T_{max}(C)\sigma]\downarrow = u$

Vérifions dans un premier temps que $\forall i, \beta_i[T_i\sigma]\downarrow = F(\beta_i)[T_i\sigma]\downarrow$. Soit $i \in [1..n]$, on effectue la preuve par récurrence sur la taille de β_i.

Cas $|\beta_i| = 1$: Dans ce cas, β_i est réduit à un axiome que l'on supposera être ax_p. Deux nouveaux cas s'offrent à nous :

– si $ax_p[T_i\sigma]\downarrow = u$, alors :

$$\begin{aligned} ax_p[T_i\sigma]\downarrow = u &\Rightarrow ax_p \in P_{\beta_1,\ldots,\beta_n}(u) \\ &\Rightarrow \min(\beta_0) \geq \min(ax_p) \\ &\quad \text{par définition de } \beta_0 \\ &\Rightarrow \beta_0[T_i\sigma]\downarrow = u \\ &\Rightarrow F(\beta_i)[T_i\sigma]\downarrow = \beta_i[T_i\sigma]\downarrow \end{aligned}$$

– si $ax_p[T_i\sigma]\downarrow \neq u$, alors $F(\beta_i) = \beta_i$ d'où $F(\beta_i)[T_i\sigma]\downarrow = \beta_i[T_i\sigma]\downarrow$.

Cas $|\beta_i| > 1$: Dans ce cas, $Top(\beta_i)$ est soit un symbole de fonction. Supposons $\beta_i = f(\alpha_1,\ldots,\alpha_m)$. Deux nouveaux cas s'offrent à nous :

– si $f(\alpha_1,\ldots,\alpha_m)[T_i\sigma]\downarrow = u$ alors :

$$\begin{aligned} f(\alpha_1,\ldots,\alpha_m)[T_i\sigma]\downarrow = u &\Rightarrow f(\alpha_1,\ldots,\alpha_m) \in P_{\beta_1,\ldots,\beta_n}(u) \\ &\Rightarrow \min(\beta_0) \geq \min(f(\alpha_1,\ldots,\alpha_m)) \\ &\quad \text{par définition de } \beta_0 \\ &\Rightarrow \beta_0[T_i\sigma]\downarrow = u \\ &\Rightarrow F(\beta_i)[T_i\sigma]\downarrow = \beta_i[T_i\sigma]\downarrow \end{aligned}$$

– si $f(\alpha_1,\ldots,\alpha_m)[T_i\sigma]\downarrow \neq u$, alors :

$$\begin{aligned} \beta_i[T_i\sigma]\downarrow &= f(\alpha_1,\ldots,\alpha_m)[T_i\sigma]\downarrow \\ &= f(\alpha_1[T_i\sigma]\downarrow,\ldots,\alpha_m[T_i\sigma]\downarrow)\downarrow \\ &= f(F(\alpha_1)[T_i\sigma]\downarrow,\ldots,F(\alpha_m)[T_i\sigma]\downarrow)\downarrow \\ &\quad \text{par hypothèse de récurrence} \\ &= f(F(\alpha_1),\ldots,F(\alpha_m))[T_i\sigma]\downarrow \\ &= F(\beta_i)[T_i\sigma]\downarrow \end{aligned}$$

On souhaite démontrer que $(\sigma, F(\beta_1), \ldots, F(\beta_n)) \in Sol(C)$. On sait déjà que $\forall i \in [1..n], F(\beta_i)[T_i\sigma]\downarrow = \beta_i[T_i\sigma]\downarrow = u_i\sigma$. Il faut donc vérifier que $(\sigma, F(\beta_1), \ldots, F(\beta_n))$ satisfait les équations de $\mathcal{E}_q$. Comme $(\sigma, \beta_1, \ldots, \beta_n) \in Sol(C)$, alors σ vérifie les équations de $\mathcal{E}_q$, ce qui est donc également le cas pour $(\sigma, F(\beta_1), \ldots, F(\beta_n))$ puisque σ ne change pas. Il reste donc à vérifier que $(\sigma, F(\beta_1), \ldots, F(\beta_n))$ vérifie les annotations de C :

On effectue une preuve par récurrence sur β_i.

Cas $|\beta_i| = 1$: Dans ce cas, β_i est réduit à un axiome que l'on supposera être ax_p. Deux cas s'offrent à nous :

- si $ax_p[T_i\sigma]\downarrow \neq u$ alors $F(ax_p) = ax_p$. β_i vérifie trivialement l'ensemble des annotations puisque $(\sigma, \beta_1, \ldots, \beta_n) \in Sol(C)$.
- si $ax_p[T_i\sigma]\downarrow = u$ alors $F(ax_p) = \beta_0$. D'après la définition des systèmes de contraintes annotés 2.7, l'ensemble des annotations $NoCons(v)$, $NoPi(v)$,$NoAx(v)$ présentent sur la i-ème contrainte sont également présente sur la $\min(\beta_0)$-ème contrainte. β_0 vérifie donc ces annotations. Il reste à vérfier pour les annotations $NoDec(v)$. Or d'après la définition des systèmes de contraintes annotés, on a : si $NoDec(v) \in Ann_i(C)$, alors $\forall j \leq i, NoDec(v) \in Ann_j(C)$. Comme $\min(\beta_0) \leq \min(\beta_i)$ d'où si une annotation $NoDec(v)$ est présente sur la i-ème contrainte, elle est également présente sur la $\min(\beta_0)$-ème contrainte d'où β_0 vérifie l'annotation $NoDec(v)$.

Cas $|\beta_i| > 1$: Supposons que $\beta_i = f(\alpha_1, \ldots, \alpha_m)$. Deux cas s'offrent à nous :

- si $f(\alpha_1, \ldots, \alpha_m)[T_i\sigma]\downarrow = u$ alors $F(\alpha_1, \ldots, \alpha_m) = \beta_0$. Pour les mêmes raisons que précédemment, β_0 vérifie l'ensemble des annotations.
- si $f(\alpha_1, \ldots, \alpha_m)[T_i\sigma]\downarrow \neq u$, alors $F(\alpha_1, \ldots, \alpha_m) = f(F(\alpha_1), \ldots, F(\alpha_m))$. Par hypothèse de récurrence, pour tout $j \in [1 \ldots m]$, $F(\alpha_j)$ vérifie les annotations. Il ne faut donc s'intéresser qu'à la valeur de f. Supposons que $f = dec$, on a donc $F(\beta_i) = dec(F(\alpha_1), F(\alpha_m))$, avec $F(\alpha_1)[T_i\sigma]\downarrow = \alpha_1[T_I\sigma]\downarrow$. Supposons maintenant par l'absurde que $F(\beta_i)$ ne vérifie pas une annotation $NoDec(v)$ alors, $F(\alpha_1)[T_i\sigma] = v\sigma$. Mais comme $F(\alpha_1)[T_i\sigma]\downarrow = \alpha_1[T_I\sigma]\downarrow$, alors $\alpha_1[T_I\sigma]\downarrow = v\sigma$. D'où β_i ne vérifie pas l'annotation $NoDec(v)$ ce qui est absurde puisque $(\sigma, \beta_1, \ldots, \beta_i) \in Sol(C)$. D'où $F(\beta_i)$ satisfait les annotations $NoDec(v)$. En suivant un raisonnement similaire, on peut démontrer que $F(\beta_i)$ vérifie également les annotations $NoAx(v), NoCons(v), NoPi(v)$.

Conclusion : On a donc bien $(\sigma, F(\beta_1), \ldots, F(\beta_n)) \in Sol(C)$

Voyons maintenant la valeur de $\|P\|_{F(\beta_1),\ldots,F(\beta_n)}$: Tout d'abord, comme on a remplacer toutes les sous-preuves ξ telle que $\xi[T_{max}(C)\sigma]\downarrow = u$ par β_0

et que $\forall \gamma \in St(\beta_0), \gamma[T_{max}(C)]\downarrow \neq u$, alors $P_{F(\beta_1),\ldots,F(\beta_n)}(u) = \{\beta_0\}$ d'où $|P_{F(\beta_1),\ldots,F(\beta_n)}| = 1$.

On va démontrer que $\forall v \in \mathcal{T}(\mathcal{C} \cup \mathcal{D} \cup \mathcal{N}), |P_{F(\beta_1),\ldots,F(\beta_n)}(v)| \leq |P_{\beta_1,\ldots,\beta_n}(v)|$:
Soit $\{\gamma_1, \ldots, \gamma_l\} = P_{\beta_1,\ldots,\beta_n}(v)$. Montrons tout d'abord que

$$\forall \xi \in P_{F(\beta_1),\ldots,F(\beta_n)}(v), \xi \in \{F(\gamma_1); \ldots; F(\gamma_l)\}$$

Soit $\xi \in P_{F(\beta_1),\ldots,F(\beta_n)}(v)$, $\exists i \in [1 \ldots n]$ tq $\xi \in St(F(\beta_i)) \wedge \xi[T_{max}(C)\sigma]\downarrow = v$. Par récurrence sur la taille de β_i, on a :

Cas $|\beta_i| = 1$: On a $\beta_i = ax_p$. Deux cas s'offrent à nous :
- Si $ax_p[T_i\sigma]\downarrow \neq u$ alors $F(ax_p) = ax_p$ d'où $\xi = ax_p$ avec $ax_p[T_i\sigma]\downarrow = v = u$ d'où $ax_p \in P_{\beta_1,\ldots,\beta_n}(v)$ d'où $\xi \in \{F(\gamma_1); \ldots; F(\gamma_l)\}$.
- Si $ax_p[T_i\sigma]\downarrow = u$ alors $F(ax_p) = \beta_0$. Comme $\beta_0 \in St(\beta_1, \ldots, \beta_n)$, alors $\xi \in P_{\beta_1,\ldots,\beta_n}(v)$. Or on sait que $\forall \beta \in St(\beta_0), \beta = F(\beta)$ d'où $F(\xi) = \xi$ et donc $\xi \in \{F(\gamma_1); \ldots; F(\gamma_l)\}$.

Cas $|\beta_i| > 1$: On a $\beta_i = f(\alpha_1, \ldots, \alpha_m)$. Deux cas s'offrent à nous :
- Si $f(\alpha_1, \ldots, \alpha_m)[T_i\sigma]\downarrow = u$ alors $F(\beta_i) = \beta_0$ et on se retrouve dans le même cas que $ax_p[T_i\sigma] = u$.
- Si $f(\alpha_1, \ldots, \alpha_m)[T_i\sigma]\downarrow \neq u$ alors $F(\beta_i) = f(F(\alpha_1), \ldots, F(\alpha_m))$. Par hypothèse de récurrence, on sait que $\forall \xi \in St(F(\alpha_j)), \xi[T_{max}(C)\sigma]\downarrow = v \Rightarrow \xi \in \{F(\gamma_1); \ldots, F(\gamma_l)\}$. Il reste donc à vérifier pour le terme $\xi = F(\beta_i)$. Or on sait que $\beta_i[T_i\sigma]\downarrow = F(\beta_i)[T_i\sigma]\downarrow$ d'où $\beta_i \in\in P_{\beta_1,\ldots,\beta_n}(v)$. Conclusion $F(\beta_i) \in \{F(\gamma_1); \ldots; F(\gamma_l)\}$.

Conclusion : $\forall \xi \in P_{F(\beta_1),\ldots,F(\beta_n)}(v), \xi \in \{F(\gamma_1); \ldots; F(\gamma_l)\}$ ce qui implique que $|P_{F(\beta_1),\ldots,F(\beta_n)}(v)| \leq |\{F(\gamma_1); \ldots; F(\gamma_l)\}|$ d'où $|P_{F(\beta_1),\ldots,F(\beta_n)}(v)| \leq |P_{\beta_1,\ldots,\beta_n}(v)|$.

Conclusion : $\forall (\sigma, \xi_1, \ldots, \xi_n) \in Sol(C), \exists (\xi'_1, \ldots, \xi'_n)$ tq $(\sigma, \xi'_1, \ldots, \xi'_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C)$. □

Théorème 3.7 (Complétude de $\mathcal{S}_{\mathcal{P}_{un}}$)**.** *Pour des systèmes de contraintes sans annotation,* $\mathcal{S}_{\mathcal{P}_{un}} \in \mathscr{S}_{comp}$.

Preuve du théorème 3.7. Soient C et C' deux systèmes de contraintes sans annotation et soient S et S' deux séquences de termes. Montrons tout d'abord l'implication droite du théorème.

$\underline{(S,C) \approx_s (S',C') \Rightarrow (S,C) \approx_s^{\mathcal{S}_\mathcal{P}} (S',C')}$: Soit $(\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C)$.

$$\left\{ \begin{array}{l} \mathcal{S}_{\mathcal{P}_{un}}(C) \subseteq Sol(C) \\ (S,C) \approx_s (S',C') \end{array} \right. \Rightarrow \exists \sigma' \text{ tq } \left\{ \begin{array}{l} (\sigma', \xi_1, ..., \xi_n) \in Sol(C') \\ S\sigma \sim S'\sigma' \end{array} \right.$$

$$\Rightarrow \exists \sigma' \text{ tq } \left\{ \begin{array}{l} (\sigma', \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C') \\ S\sigma \sim S'\sigma' \end{array} \right.$$

par le lemme 3.5

D'où $\forall (\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C), \exists \sigma' \text{ tq } (\sigma', \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C') \wedge S\sigma \sim S'\sigma'$. Par symétrie, on démontre que $\forall (\sigma', \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C'), \exists \sigma \text{ tq } (\sigma, \xi_1, ..., \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C) \wedge S\sigma \sim S'\sigma'$.

Conclusion : $(S,C) \approx_s^{\mathcal{S}_\mathcal{P}} (S',C')$.

On montre maintenant l'implication gauche du théorème :

$\underline{(S,C) \approx_s (S',C') \Leftarrow (S,C) \approx_s^{\mathcal{S}_\mathcal{P}} (S',C')}$:

Soit $(\sigma, \xi_1, \ldots, \xi_n) \in Sol(C)$:

$$(\sigma, \xi_1, \ldots, \xi_n) \in Sol(C) \Rightarrow \exists (\xi'_1, \ldots, \xi'_n) \text{ tq } (\sigma, \xi'_1, \ldots, \xi'_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C)$$

par le lemme 3.6

$$\Rightarrow \exists \sigma' \text{ tq } (\sigma', \xi'_1, \ldots, \xi'_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C') \wedge S\sigma \sim S'\sigma'$$

car $(S,C) \approx_s^{\mathcal{S}_\mathcal{P}} (S',C')$

Or $\forall i, \xi_i[T_i\sigma]\downarrow = \xi'_i[T_i\sigma]\downarrow$ puisque même solution du premier ordre. Comme $S\sigma \sim S'\sigma' \Rightarrow T_i\sigma \sim T'_i\sigma'$, on a $\forall i, \xi_i[T'_i\sigma']\downarrow = \xi'_i[T'_i\sigma']\downarrow$. D'où $(\sigma', \xi_1, \ldots, \xi_n) \in Sol(C')$ puisque C et C' sont sans annotations.

On a donc $\forall (\sigma, \xi_1, \ldots, \xi_n) \in Sol(C), \exists \sigma' \text{ tq } (\sigma', \xi_1, \ldots, \xi_n) \in Sol(C')$. Par symétrie, on démontre que $\forall (\sigma', \xi_1, \ldots, \xi_n) \in Sol(C'), \exists \sigma \text{ tq } (\sigma, \xi_1, \ldots, \xi_n) \in Sol(C)$.

Conclusion : $(S,C) \approx_s (S',C')$. □

C.2 Complétude et correction des règles de réduction

Un des gros problèmes à gérer est l'introduction des annotations. Ces annotations sont très utiles car elles permettent de restreindre explicitement les solutions à considérer mais d'un autre coté, elles ne doivent pas être disposées n'importe comment sur un système de contraintes. Nous définissons donc la propriété suivante sur les couples $(S,C), (S',C')$ qui devra toujours être vérifiée.

Définition C.3 (Couples système de contraintes/séquencs adéquats)**.** Soient $C = ([T_1 \Vdash u_1, \ldots, T_n \Vdash u_n], \mathcal{E}_q)$ et $C' = ([T'_1 \Vdash u'_1, \ldots, T'_n \Vdash u'_n], \mathcal{E}'_q)$ deux systèmes de contraintes et soient S et S' deux séquences de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, X_v)$. On dit que $(S, C), (S', C')$ sont *adéquats* ssi $\forall \sigma, \sigma', \forall (\xi_1, \ldots, \xi_n) \in \Pi^n$ tq :

- $S\sigma \sim S'\sigma'$
- $\forall i \in [1..n], \xi_i[T_i\sigma]{\downarrow} = u_i\sigma \wedge \xi_i[T'_i\sigma']{\downarrow} = u'_i\sigma'$
- σ satisfait $\mathcal{E}_q$ et σ' satisfait $\mathcal{E}'_q$

alors $\forall i \in [1..n], \forall p \in \mathbb{N}$,

1. pour toute annotation $NoX(u) \in Ann_{i,p}(C)$ non satisfaite par $(\sigma, \xi_1, \ldots, \xi_n)$, il existe une annotation $NoX(u') \in Ann_{i,p}(C')$ non satisfaite par $(\sigma', \xi_1, \ldots, \xi_n)$.
2. pour tout annotation $NoX(u') \in Ann_{i,p}(C')$ non sastisfaite par $(\sigma', \xi_1, \ldots, \xi_n)$, il existe une annotation $NoX(u) \in Ann_{i,p}(C)$ non satisfaite par $(\sigma, \xi_1, \ldots, \xi_n)$.

Remarquons que deux systèmes de contraintes sans annotation vérifient trivialement ces propriétés. Nous montrerons de plus que la propriété d'adéquation est invariante par application de nos règles, si bien que nous n'avons à considérer que des systèmes adéquats.

De plus, bien que les propriétés énoncées dans la définition C.3 soient assez longues, elle ne font qu'énoncer une symétrie entre les annotations sur les deux systèmes de contraintes.

Exemple C.2. Reprenons les systèmes de contraintes définies dans l'exemple 3.3. Prenons $\sigma = \sigma' = Id$ et $(\xi_1, \xi_2) = (ax_1, \langle ax_2, ax_3 \rangle)$. On a bien les propriétés :

- $S\sigma \sim S'\sigma'$
- $\xi_1[S\sigma]{\downarrow} = \langle a, b \rangle = u_1\sigma$ et $\xi_1[S'\sigma']{\downarrow} = \langle a, b \rangle = u'_1\sigma'$
- $\xi_2[S\sigma]{\downarrow} = \langle a, b \rangle = u_2\sigma$ et $\xi_2[S'\sigma']{\downarrow} = \langle a, b \rangle = u'_2\sigma'$
- σ et σ' vérifient les équations de $\mathcal{E}_q$ et $\mathcal{E}'_q$ (trivial vu qu'ils sont vides).

Pourtant, on le terme $a^{F'}$ sur la deuxième contrainte de C' avec $NoAx(a) \in F'$. Donc d'après la propriété, il devrait y avoir également sur le terme a^F de la deuxième contrainte de C, une annotation $NoAx(u) \in F$ telle que $u\sigma = a\sigma$, ce qui n'est pas le cas. Les deux couples $(S, C), (S', C')$ ne sont donc pas adéquats.

Lemme 4.1. *Soient C, C' deux systèmes de contraintes et S, S' deux séquences de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$ avec $(S, C), (S', C')$ adéquats. On a :* $\forall \sigma, \sigma', \forall (\xi_1, \ldots, \xi_n) \in \Pi^n, \forall (\xi'_1, \ldots, \xi'_n) \in \Pi^n$, *si :*

- $S\sigma \sim S'\sigma'$
- $\forall i \in [1..n], \xi_i[T_i\sigma]\downarrow = u_i\sigma \wedge \xi_i[T'_i\sigma']\downarrow = u'_i\sigma'$
- σ *vérifie* $\mathcal{E}_q$ *et* σ' *vérifie* $\mathcal{E}'_q$

alors on a :

$$(\sigma, \xi'_1, \ldots, \xi'_n) \in Sol(C) \Leftrightarrow (\sigma', \xi'_1, \ldots, \xi'_n) \in Sol(C')$$

Preuve du lemme 4.1. Soient $C = ([T_1 \Vdash u_1^{F_1}; \ldots, T_n \Vdash u_n^{F_n}], \mathcal{E}_q), C' = ([T'_1 \Vdash u_1'^{F'_1}; \ldots, T'_n \Vdash u_n'^{F'_n}], \mathcal{E}'_q)$ deux systèmes de contraintes et S, S' deux séquences de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$ avec $(S, C), (S', C')$ adéquats. On a : $\forall \sigma, \sigma', \forall (\xi_1, \ldots, \xi_n) \in \Pi^n$ tq :

- $S\sigma \sim S'\sigma'$
- $\forall i \in [1..n], \xi_i[T_i\sigma]\downarrow = u_i\sigma \wedge \xi_i[T'_i\sigma']\downarrow = u'_i\sigma'$
- σ vérifie $\mathcal{E}_q$ et σ' vérifie $\mathcal{E}'_q$

On va montrer un seul sens de l'implication (l'autre sens s'effectuant par symétrie) :

Supposons que $(\sigma, \xi'_1, \ldots, \xi'_n) \in Sol(C)$, on va démontrer que $(\sigma', \xi'_1, \ldots, \xi'_n) \in Sol(C')$.

$$\begin{cases} \forall i \in [1..n], \xi_i[T_i\sigma]\downarrow = u_i\sigma \\ (\sigma, \xi'_1, \ldots, \xi'_n) \in Sol(C) \end{cases} \Rightarrow \forall i \in [1 \ldots n], \xi_i[T_i\sigma]\downarrow = \xi'_i[T_i\sigma]\downarrow$$

$$\Rightarrow \forall i \in [1 \ldots n], \xi_i[T'_i\sigma']\downarrow = \xi'_i[T'_i\sigma']\downarrow$$

$$\text{car } S\sigma \sim S'\sigma'$$

$$\Rightarrow \forall i \in [1 \ldots n], \xi'_i[T'_i\sigma']\downarrow = u'_i\sigma'$$

De plus, on sait que σ' vérifie $\mathcal{E}'_q$. Il reste donc à vérifier $(\sigma', \xi'_1, \ldots, \xi'_n)$ vérifie bien les annotations de C'.

Supposons par l'absurde que $(\sigma', \xi'_1, \ldots, \xi'_n)$ ne vérifie pas une des annotations de C'. D'après la définition des systèmes adéquats C.3, cela implique qu'il existe une annotation dans C qui n'est pas satisfaite par $(\sigma, \xi'_1, \ldots, \xi'_n)$ ce qui est absurde puisque $(\sigma, \xi'_1, \ldots, \xi'_n) \in Sol(C)$.

Conclusion : $(\sigma', \xi'_1, \ldots, \xi'_n) \in Sol(C')$

□

Dans la partie principale du rapport, nous avons énoncé une définition des règles adéquates. Néanmoins, pour un soucis de place, nous n'avons pu en donner qu'une version simplifié. Dans cette section, on va redonner la définition complète de règles adéquates ainsi que l'énoncé du théorème 4.2. Sur la figure 1, (S_1, C_1) est la simplification de (S, C) restreint aux solutions $\mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C)$. De même, (S'_1, C'_1) est la simplification de (S', C') restreint aux solutions $\mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C')$. Néanmoins, toute simplification n'est pas possible (sinon on simplifierait toujours vers $\bot$). C'est pourquoi, chaque simplification est paramétrée par deux fonctions F et G telles que si C, C' ont n contraintes et C_1, C'_1 ont m contraintes, alors

$$F : \Pi^n \to \Pi^m \qquad G : \Pi^m \to \Pi^n$$

Définition C.4 (Simplification adéquates)**.** Soit une simplification de $((S, C), (S', C'))$ restreint aux support $\mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C)$ et $\mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}'}(C')$ vers $((S_1, C_1), (S'_1, C'_1))$ définie par F et G. Supposons que n est le nombre de contraintes de C, C' et m celui de C_1, C'_1. F et G sont *adéquates par rapport à* $\mathcal{P}$ ssi elles vérifient les propriétés suivantes :

1. $F(\mathcal{S}_{2,\mathcal{P}_{un} \wedge \mathcal{P}}(C)) \subseteq Sol_2(C_1)$ et $F(\mathcal{S}_{2,\mathcal{P}_{un} \wedge \mathcal{P}}(C')) \subseteq Sol_2(C'_1)$
2. $G(\mathcal{S}_{2,\mathcal{P}_{un}}(C_1)) \subseteq Sol_2(C)$ et $G(\mathcal{S}_{2,\mathcal{P}_{un}}(C'_1)) \subseteq Sol_2(C')$
3. $\forall (\xi_1, \ldots, \xi_m) \in \mathcal{S}_{2,\mathcal{P}_{un}}(C_1) \cup \mathcal{S}_{2,\mathcal{P}_{un}}(C'_1)$,

$$\begin{gathered} G(\xi_1, \ldots, \xi_m) \in Sol_2(C) \cap Sol_2(C') \wedge S\sigma \sim S'\sigma' \\ \Rightarrow \\ (\xi_1, \ldots, \xi_m) \in Sol_2(C_1) \cap Sol_2(C'_1) \wedge S_1\alpha \sim S'_1\alpha' \end{gathered}$$

   avec $\sigma = \lambda_C \circ G(\xi_1, \ldots, \xi_m)$ et $\sigma' = \lambda_{C'} \circ G(\xi_1, \ldots, \xi_m)$
   et $\alpha = \lambda_{C_1}(\xi_1, \ldots, \xi_m)$ et $\alpha' = \lambda_{C'_1}(\xi_1, \ldots, \xi_m)$.
4. $\forall (\xi_1, \ldots, \xi_n) \in \mathcal{S}_{2,\mathcal{P}_{un} \wedge \mathcal{P}}(C) \cup \mathcal{S}_{2,\mathcal{P}_{un} \wedge \mathcal{P}}(C')$,

$$\begin{gathered} F(\xi_1, \ldots, \xi_n) \in Sol_2(C_1) \cap Sol_2(C'_1) \wedge S_1\alpha \sim S'_1\alpha \\ \Rightarrow \\ (\xi_1, \ldots, \xi_n) \in Sol_2(C) \cap Sol_2(C') \wedge S\sigma \sim S'\sigma' \end{gathered}$$

   avec $\sigma = \lambda_C(\xi_1, \ldots, \xi_n)$ et $\sigma' = \lambda_{C'}(\xi_1, \ldots, \xi_n)$
   et $\alpha = \lambda_{C_1} \circ F(\xi_1, \ldots, \xi_n)$ et $\alpha' = \lambda_{C'_1} \circ F(\xi_1, \ldots, \xi_n)$.

Définition C.5 (Règles adéquates)**.** Soient C, C' deux systèmes de contraintes et soient S, S' deux séquences de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$. Une règle de réduction de contraintes R est dite *adéquate* par rapport à $(S, C), (S', C')$ si elle suit le modèle de la figure 1 et si :

– F_1 et G_1 sont adéquates par rapport à $\mathcal{P}$.
– F_2 et G_2 sont adéquates par rapport à $\neg\mathcal{P}$.
– $(S,C),(S',C')$ adéquats $\Rightarrow \left\{ \begin{array}{l} (S_1,C_1),(S'_1,C'_1) \text{ adéquats} \\ (S_2,C_2),(S'_2,C'_2) \text{ adéquats} \end{array} \right.$

Théorème C.1 (Complétude et correction des règles de réduction)**.** *Soient C et C' deux systèmes de contraintes et soient S et S' deux séquences de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$ tels que $(S,C),(S',C')$ soient adéquats. Soit R une règle de réduction de contraintes adéquate par rapport à $(S,C),(S',C')$. Alors, pour tout $((S,C),(S',C')) \rightsquigarrow^R \{((S_1,C_1),(S'_1,C'_1));((S_2,C_2),(S'_2,C'_2))\}$,*

$$(S,C) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S',C') \Leftrightarrow \left\{ \begin{array}{l} (S_1,C_1) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S'_1,C'_1) \\ (S_2,C_2) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S'_2,C'_2) \end{array} \right.$$

avec $(S_1,C_1),(S'_1,C'_1)$ adéquats et $(S_2,C_2),(S'_2,C'_2)$ adéquats.

Preuve du théorème C.1. Soient $C = ([T_1 \Vdash u_1^{F_1}, \ldots, T_n \Vdash u_n^{F_n}], \mathcal{E}_q)$ et $C' = ([T'_1 \Vdash {u'_1}^{F'_1}, \ldots, T'_n \Vdash {u'_n}^{F'_n}], \mathcal{E}'_q)$ deux systèmes de contraintes et soient S et S' deux séquences de termes de $\mathcal{T}(\mathcal{C} \cup \mathcal{N}, \mathcal{X}_v)$ tels que $(S,C),(S',C')$ sont adéquats. Soit R une règle de réduction de contraintes adéquate par rapport à $(S,C),(S',C')$. Soient $(S_1,C_1),(S'_1,C'_1)$ et $(S_2,C_2),(S'_2,C'_2)$ tels que :

$$((S,C),(S',C')) \rightsquigarrow^R \{((S_1,C_1),(S'_1,C'_1));((S_2,C_2),(S'_2,C'_2))\}$$

Montrons tout d'abord l'implication droite $\Rightarrow$:

Supposons que $(S,C) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S',C')$. Soit $(\sigma,\xi_1,\ldots,\xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C_1)$. Comme F_1 et G_1 sont adéquates par rapport à $\mathcal{P},\mathcal{P}'$, alors :

$$\begin{array}{ll} (\sigma,\xi_1,\ldots,\xi_m) \in \mathcal{S}_{\mathcal{P}_{un}}(C_1) \Rightarrow & (\sigma, G_1(\xi_1,\ldots,\xi_m)) \in Sol(C) \\ & \quad \text{car prop. 2 de la def. C.4} \\ & \Rightarrow \exists (\xi'_1,\ldots,\xi'_n) \in \Pi^n \text{ tq } (\sigma,\xi'_1,\ldots,\xi'_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C) \\ & \quad \text{car th. 3.6} \\ & \Rightarrow \exists \sigma' \text{ tq } (\sigma',\xi'_1,\ldots,\xi'_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C') \wedge S\sigma \sim S'\sigma' \\ & \quad \text{car } (S,C) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S',C') \\ & \Rightarrow (\sigma', G_1(\xi_1,\ldots,\xi_m)) \in Sol(C') \\ & \quad \text{par le lemme 4.1 et } (S,C),(S',C') \text{ adéquats} \end{array}$$

Rappelons rapidement les résultats obtenus jusque là :

– $(\sigma, \xi_1, \ldots, \xi_m) \in \mathcal{S}_{\mathcal{P}_{un}}(C_1)$
– $(\sigma, G(\xi_1, \ldots, \xi_m)) \in Sol(C)$
– $(\sigma', G(\xi_1, \ldots, \xi_m)) \in Sol(C')$
– $S\sigma \sim S'\sigma'$

Donc d'après la propriété 3 de la définition C.4, on a $S_1\sigma \sim S'_1\sigma$ et $(\sigma', \xi_1, \ldots, \xi_m) \in Sol(C'_1)$. Le lemme 3.5 permet de conclure que $(\sigma, \xi_1, \ldots, \xi_m) \in \mathcal{S}_{\mathcal{P}_{un}}(C_1)$ implique $(\sigma', \xi_1, \ldots, \xi_m) \in \mathcal{S}_{\mathcal{P}_{un}}(C'_1)$.

On a donc : $\forall (\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C_1), \exists \sigma'$ tq $(\sigma', \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C'_1) \wedge S_1\sigma \sim S'_1\sigma'$. Par symétrie sur (S, C),(S', C'), et sur $(S_1, C_1), (S'_1, C'_1)$, on montre que $\forall (\sigma', \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C'_1), \exists \sigma$ tq $(\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C_1) \wedge S_1\sigma \sim S'_1\sigma'$.

D'où $(S, C) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S', C') \Rightarrow (S_1, C_1) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S'_1, C'_1)$

On utilise exactement la même preuve pour démontrer que $(S, C) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S', C') \Rightarrow (S_2, C_2) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S'_2, C'_2)$. Il suffit de remplacer $\mathcal{P}, \mathcal{P}'$ par $\neg\mathcal{P}, \neg\mathcal{P}'$, $(S_1, C_1), (S'_1, C'_1)$ par $(S_2, C_2), (S'_2, C'_2)$ et enfin F_1, F_1^{-1} par F_2, F_2^{-1}.

Conclusion :

$$(S, C) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S', C') \Rightarrow \begin{cases} (S_1, C_1) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S'_1, C'_1) \\ (S_2, C_2) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S'_2, C'_2) \end{cases}$$

Montrons maintenant l'implication gauche $\Leftarrow$:

Supposons que $(S_1, C_1) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S'_1, C'_1)$ et $(S_2, C_2) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S'_2, C'_2)$.

Soit $(\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C)$. Supposons dans un premier temps que $(\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C)$.

$$\begin{aligned}
(\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C) &\Rightarrow (\sigma, F_1(\xi_1, \ldots, \xi_n)) \in Sol(C_1) \\
&\quad \text{(prop. 1 de la def. C.4)} \\
&\Rightarrow \exists (\xi'_1, \ldots, \xi'_n) \in \Pi^n \text{ tq } (\sigma, \xi'_1, \ldots, \xi'_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C_1) \\
&\quad \text{car th. 3.6} \\
&\Rightarrow \exists \sigma' \text{ tq } (\sigma', \xi'_1, \ldots, \xi'_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C'_1) \\
&\quad \wedge S_1\sigma \sim S'_1\sigma' \\
&\quad \text{car } (S_1, C_1) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S'_1, C'_1) \\
&\Rightarrow (\sigma', F_1(\xi_1, \ldots, \xi_n)) \in Sol(C'_1) \\
&\quad \text{par le lemme 4.1 et } (S, C), (S', C') \text{ adéquats}
\end{aligned}$$

Rappelons rapidement les résultats obtenus jusque là :
- $(\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C)$
- $(\sigma, F_1(\xi_1, \ldots, \xi_n)) \in Sol(C_1)$
- $(\sigma', F_1(\xi_1, \ldots, \xi_n)) \in Sol(C'_1)$
- $S_1\sigma \sim S'_1\sigma'$

On peut donc appliquer la propriété 4 de la définition C.4 d'où $S\sigma \sim S'\sigma'$ et $(\sigma', \xi_1, \ldots, \xi_n) \in Sol(C')$. Le lemme 3.5 permet de conclure que $(\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C)$ implique $(\sigma', \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C')$.

On a donc $\forall (\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C) \Rightarrow \exists \sigma'$ tq $(\sigma', \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C') \wedge S\sigma \sim S'\sigma'$.

Une preuve symétrique en remplaçant $\mathcal{P}$ par $\neg\mathcal{P}$, $(S_1, C_1), (S'_1, C'_1)$ par $(S_2, C_2), (S'_2, C'_2)$ et enfin F_1, F_1^{-1} par F_2, F_2^{-1} nous permet de démontrer que $\forall (\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un} \wedge \neg\mathcal{P}}(C) \Rightarrow \exists \sigma'$ tq $(\sigma', \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C') \wedge S\sigma \sim S'\sigma'$.

Comme $\mathcal{S}_{\mathcal{P}_{un}}(C) = \mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C) \uplus \mathcal{S}_{\mathcal{P}_{un} \wedge \neg\mathcal{P}}(C)$, on a donc : $\forall (\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C), \exists \sigma'$ tq $(\sigma', \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C') \wedge S\sigma \sim S'\sigma'$.

Par symétrie sur $(S, C), (S', C')$, la preuve précédente nous permet de montrer que $\forall (\sigma', \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}} C', \exists \sigma$ tq $(\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un}}(C) \wedge S\sigma \sim S'\sigma'$

Conclusion :

$$\begin{cases} (S_1, C_1) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S'_1, C'_1) \\ (S_2, C_2) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S'_2, C'_2) \end{cases}$$
$$\Rightarrow$$
$$(S, C) \approx_s^{\mathcal{S}_{\mathcal{P}_{un}}} (S', C')$$

□

C.3 Vérification des propriétés des règles

Il nous faut démonter que pour l'ensemble des règles énoncées dans l'annexe A.2, pour tout $(S, C), (S', C')$ vérifiant les conditions de R, R est adéquates par rapport à $(S, C), (S', C')$. Un des gros problèmes de ces preuves est qu'elles sont faites au cas par cas et comme de nombreuses règles ont été énoncées, la taille de cette section est vite devenu trop imposante (environ 40-50 pages de preuves) et les preuves en elles-mêmes sont devenues très répétitives. Ce problème vient du fait que nous avons dans un premier temps établi des règles qui collent parfaitement aux prédicats cryptographiques utilisés. En ce sens, nous avons préféré créer de nombreuses règles simples,

compréhensibles et qui sont correctes sans essayer de les généraliser au maximum. Cela a pour conséquence qu'au sein des preuves, de nombreuses choses se répètent. Un des objectifs futurs sera donc de réduire le nombre de règles utilisés et de les généraliser.

Néanmoins, pour donner un aperçu de ce à quoi ressemble ces preuves, nous allons donner la preuve de vérification de la règle R_1.

Preuve : Vérification de la règle R_1. Soit i la contrainte sur laquelle la règle R_1 s'applique. Définissons dans un premier le prédicat de cette règle :

$$\mathcal{P} : Top(\xi_i) = f$$

Voyons maintenant les fonctions F_1, G_1, F_2, G_2 :

- $F_1 : \Pi^n \to \Pi^{n+1}$ avec $F_1(\xi_1, \ldots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \ldots, \xi_n) = (\xi_1, \ldots, \xi_{i-1}, \alpha_1, \alpha_2, \xi_{i+1}, \ldots, \xi_n)$.
- $G_1 : \Pi^{n+1} \to \Pi^n$ avec $G_1(\xi_1, \ldots, \xi_{i-1}, \alpha_1, \alpha_2, \xi_{i+1}, \ldots, \xi_n) = (\xi_1, \ldots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \ldots, \xi_n)$.
- $F_2 = Id$
- $G_2 = Id$

Démontrons dans un premier temps que $F(\mathcal{S}_{2,\mathcal{P}_{un} \wedge \mathcal{P}}(C)) \subseteq Sol_2(C_1)$: Soit $(\sigma, \xi_1, \ldots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \ldots, \xi_n) \in \mathcal{S}_{\mathcal{P}_{un} \wedge \mathcal{P}}(C)$, puisque $f \in \mathcal{C}$, on a :

$$\begin{aligned} f(\alpha_1, \alpha_2)[T_i\sigma]\downarrow = f(t_1, t_2)\sigma &\Rightarrow f(\alpha_1, \alpha_2)[T_i\sigma]\downarrow = f(t_1\sigma, t_2\sigma) \\ &\Rightarrow \alpha_1[T_i\sigma]\downarrow = t_1\sigma \wedge \alpha_2[T_i\sigma]\downarrow = t_2\sigma \end{aligned} \tag{2}$$

D'où σ solution du premier ordre de $(\xi_1, \ldots, \xi_{i-1}, \alpha_1, \alpha_2, \xi_{i+1}, \ldots, \xi_n)$. On va vérifier que σ vérifie les équations rajoutées $\{f(v_1, v_2) \neq f(t_1, t_2) \mid f(v_1, v_2) \in E\}$. Soit $f(v_1, v_2) \in E$. On a :

$$\begin{aligned} f(v_1, v_2) \in E &\Rightarrow NoCons(f(v_1, v_2)) \in F_i \\ &\Rightarrow \forall j \in [1..n], \forall f(\xi_b, \xi_c) \in St(\xi_j), f(\xi_b, \xi_c)[T_j\sigma]\downarrow \neq f(v_1, v_2)\sigma \\ &\Rightarrow f(\alpha_1, \alpha_2)[T_i\sigma]\downarrow \neq f(v_1, v_2)\sigma \\ &\Rightarrow f(t_1, t_2)\sigma \neq f(v_1, v_2)\sigma \end{aligned} \tag{3}$$

Donc d'après 3, σ vérifie bien les équations de C_1.

Il reste donc à vérifier que $(\sigma, F(\xi_1, \ldots, \xi_n))$ vérifie les annotations de C_1. L'ensemble des annotations déjà présente dans C et qui sont reproduites

dans C_1 sont trivialement vérifiées puisqu'aucune nouvelle sous-preuve n'est créée. Conclusion : $(\sigma, F(\xi_1, \ldots, \xi_n)) \in Sol(C_1)$.

Par symétrie, on démontre également que $F(\mathcal{S}_{2,\mathcal{P}_{un}\wedge\mathcal{P}}(C')) \subseteq Sol(C'_1)$.

Démontrons maintenant que $G(Sol_2(C_1)) \subseteq Sol_2(C)$: Soit $(\sigma, \xi_1, \ldots, \xi_{i-1}, \alpha_1, \alpha_2, \xi_{i+1}, \ldots, \xi_n) \in Sol_2(C_1)$. On a :

$$\begin{array}{ll} \left\{ \begin{array}{l} \alpha_1[T_i\sigma]\downarrow = t_1\sigma \\ \alpha_2[T_i\sigma]\downarrow = t_2\sigma \end{array} \right. & \Rightarrow f(\alpha_1[T_i\sigma]\downarrow, \alpha_2[T_i\sigma]\downarrow) = f(t_1\sigma, t_2\sigma) \\ & \Rightarrow f(\alpha_1[T_i\sigma]\downarrow, \alpha_2[T_i\sigma]\downarrow) = f(t_1, t_2)\sigma \\ & \Rightarrow f(\alpha_1, \alpha_2)[T_i\sigma]\downarrow = f(t_1, t_2)\sigma \\ & \quad \text{car } f \in \mathcal{C} \end{array}$$

D'où σ solution du premier ordre de $(\xi_1, \ldots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \ldots, \xi_n)$. σ vérifie trivialement les équations $\mathcal{E}_q$ de C puisque $\mathcal{E}_q \subseteq \mathcal{E}_q^1$. Il ne reste plus qu'à vérifier $(\sigma, \xi_1, \ldots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \ldots, \xi_n)$ vérifie les annotations de C. Or la seule sous-preuve que l'on a créé est $f(\alpha_1, \alpha_2)$ et aucune annotation n'est rajouté en passant de C_1 à C. Donc si $(\sigma, \xi_1, \ldots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \ldots, \xi_n)$ ne vérifie pas les annotations, cela ne peut être dû qu'à la preuve $f(\alpha_1, \alpha_2)$. Supposons qu'il existe $NoCons(f(v_1, v_2)) \in F_i$ tq $f(\alpha_1, \alpha_2)[T_i\sigma]\downarrow = f(v_1, v_2)\sigma$ et raisonnons par l'absurde. On a :

$$\begin{array}{rl} f(\alpha_1, \alpha_2)[T_i\sigma]\downarrow = f(v_1, v_2)\sigma & \Rightarrow f(t_1, t_2)\sigma = f(v_1, v_2)\sigma \\ & \Rightarrow \exists mgu(f(t_1, t_2), f(v_1, v_2)) \\ & \Rightarrow f(v_1, v_2) \in E \end{array}$$

Or σ vérifie les équations de $\mathcal{E}_q^1 = \mathcal{E}_q \cup \{f(v_1, v_2) \neq f(t_1, t_2) \mid f(v_1, v_2) \in E\}$ d'où :

$$f(v_1, v_2)\sigma \neq f(t_1, t_2)\sigma$$

Il y a bien une absurdité d'où $(\sigma, \xi_1, \ldots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \ldots, \xi_n)$ vérifie les annotations de C.

Conclusion : $G(Sol_2(C_1)) \subseteq Sol_2(C)$.

Par symétrie, on démontre également que $G(Sol_2(C'_1)) \subseteq Sol_2(C')$.

On va maintenant démontrer que si $(S, C), (S', C')$ sont adéquats alors $(S_1, C_1), (S'_1, C'_1)$ sont adéquats également : Soient σ, σ' et $(\xi_1, \ldots, \xi_{i-1}, \alpha_1, \alpha_2, \xi_{i+1}, \ldots, \xi_n) \in \Pi^{n+1}$ tq :

– $S_1\sigma \sim S'_1\sigma'$
– $\forall j \in [1 \dots i-1] \cup [i+1 \dots n], \xi_j[T_j\sigma]\downarrow = u_j\sigma \wedge \xi_j[T'_j\sigma']\downarrow = u'_j$
– $\alpha_1[T_i\sigma]\downarrow = t_1\sigma \wedge \alpha_1[T'_i\sigma']\downarrow = t'_1\sigma'$
– $\alpha_2[T_i\sigma]\downarrow = t_2\sigma \wedge \alpha_2[T'_i\sigma']\downarrow = t'_2\sigma'$
– σ vérifie $\mathcal{E}_q^1$ et σ' vérifie $\mathcal{E}_q^{1'}$

Supposons qu'une annotation $NoX(u)$ n'est pas satisfaite par $(\sigma, \xi_1, \dots, \xi_{i-1}, \alpha_1, \alpha_2, \xi_{i+1}, \dots, \xi_n)$ alors $NoAx(u)$ n'est également pas satisfaite par $(\sigma, \xi_1, \dots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \dots, \xi_n)$ puisqu'on ne supprime aucune sous-preuves. Or comme $(S, C), (S', C')$ sont adéquats, cela implique qu'il existe une annotation $NoX(u')$ qui n'est pas satisfait par $(\sigma, \xi_1, \dots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \dots, \xi_n)$. Soit $NoX(u')$ n'est pas satisfait à cause d'un sous-terme de $(\xi_1, \dots, \xi_{i-1}, \alpha_1, \alpha_2, \xi_{i+1}, \dots, \xi_n)$ soit à cause de $f(\alpha_1, \alpha_2)$. Dans le premier cas, on aura bien que $NoAx(u')$ n'est pas satisfait par $(\sigma', \xi_1, \dots, \xi_{i-1}, \alpha_1, \alpha_2, \xi_{i+1}, \dots, \xi_n)$ ce qui vérifie la propriété des systèmes adéquats. Dans le second cas, seul une annotation $NoCons(u')$ peut en être le cause. Or on a vu précédemment que dû aux équations ainsi qu'au fait que $NoCons(f(t_1, t_2)) \notin F_i$, une telle annotation $NoCons(u')$ ne pouvait exister. On a donc bien que les systèmes $(S, C), (S', C')$ sont adéquats.

On va maintenant démontrer la propriété 3 de la définition C.4 : Supposons que :

$$(\sigma, \xi_1, \dots, \xi_{i-1}, \alpha_1, \alpha_2, \xi_{i+1}, \dots, \xi_{n+1}) \in \mathcal{S}_{\mathcal{P}_{un}}(C_1)$$
$$(\sigma', \xi_1, \dots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \dots, \xi_{n+1}) \in Sol(C')$$
$$S\sigma \sim S'\sigma'$$

Démontrer $S_1\sigma \sim S'_1\sigma'$ est trivial puisque $S = S_1$ et $S' = S'_1$. Il reste donc à montrer que $(\sigma', \xi_1, \dots, \xi_{i-1}, \alpha_1, \alpha_2, \xi_{i+1}, \dots, \xi_{n+1}) \in Sol(C'_1)$. Comme $(\sigma', \xi_1, \dots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \dots, \xi_{n+1}) \in Sol(C')$, alors :

$$\begin{aligned} f(\alpha_1, \alpha_2)[T'_i\sigma']\downarrow &= f(t'_1, t'_2)\sigma' \\ &= f(t_1\sigma, t_2\sigma) \end{aligned}$$

D'où $\alpha_1[T'_i\sigma']\downarrow = t'_1\sigma'$ et $\alpha_2[T'_i\sigma']\downarrow = t'_2\sigma'$. De plus, l'ensemble des annotations $NoCons(f(v'_1, v'_2)) \in F_i$ sont vérifiées par $(\sigma', \xi_1, \dots, \xi_{i-1}, f(\alpha_1, \alpha_2), \xi_{i+1}, \dots, \xi_{n+1})$ d'où $\forall f(v'_1, v'_2) \in E', f(v'_1, v'_2)\sigma' \neq f(t'_1, t'_2)$. De plus, σ' vérifie les équations de $\mathcal{E}'_q$ d'où σ' vérifie les équations de $\mathcal{E}_q^{1'}$. Il reste à vérifier que les annotations sont vérifiées. Or $(S_1, C_1), (S'_1, C'_1)$ sont adéquats. Donc d'après le lemme 4.1 :

$$(\sigma', \xi_1, \dots, \xi_{i-1}, \alpha_1, \alpha_2, \xi_{i+1}, \dots, \xi_{n+1}) \in Sol(C'_1)$$

La démonstration de la propriété 4 de la définition C.4 est similaire à celle de la propriété 3 (utilisation du lemme 4.1).

On a démontrer l'ensemble des propriétés pour F_1,G_1. Il faut encore démontrer les mêmes propriétés pour F_2 et G_2 sont adéquats par rapport à $\neg\mathcal{P}$. Démontrons en premier lieu que $F_2(\mathcal{S}_{2,\mathcal{P}_{un}\wedge\neg\mathcal{P}}(C)) \subseteq Sol(C_2)$.

Soit $(\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{2,\mathcal{P}_{un}\wedge\neg\mathcal{P}}(C)$ d'où $Top(\xi_i) \neq f$. De plus, comme $(\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{2,\mathcal{P}_{un}}(C)$, alors $\forall j \in [1 \ldots n], \forall \xi_a \in \Pi_c, \forall \xi_b \in \Pi, \xi_j = \xi_a\lfloor\xi_b\rfloor \wedge \xi_b[T_i\sigma]\downarrow = f(t_1, t_2)\sigma \Rightarrow \xi_b = \xi_i$. D'où $(\sigma, \xi_1, \ldots, \xi_n)$ vérifie $NoCons(f(t_1, t_2))$ et donc $(\sigma, \xi_1, \ldots, \xi_n) \in \mathcal{S}_{2,\mathcal{P}_{un}}(C_2)$.

Conclusion : $F_2(\mathcal{S}_{2,\mathcal{P}_{un}\wedge\neg\mathcal{P}}(C)) \subseteq \mathcal{S}_{2,\mathcal{P}_{un}}(C_2)$.
Par symétrie, on a : $F_2(\mathcal{S}_{2,\mathcal{P}_{un}\wedge\neg\mathcal{P}}(C')) \subseteq \mathcal{S}_{2,\mathcal{P}_{un}}(C'_2)$.

Les propriétés 2,3,4 son triviales et reviennent à appliquer le lemme 4.1. On va donc d'attarder à démontrer que $(S_2, C_2), (S'_2, C'_2)$ sont adéquats. Les annotations déjà présentes dans C et C' sont trivialement vérifiés. Voyons le cas de $NoCons(f(t_1, t_2))$ et celui de $NoCons(f(t'_1, t'_2))$.
Supposons qu'il existe $(\alpha_1, \alpha_2) \in \Pi^2$ tel que $f(\alpha_1, \alpha_2)[S\sigma]\downarrow = f(t_1, t_2)\sigma$ d'où

$$\begin{aligned} f(\alpha_1, \alpha_2)[S\sigma]\downarrow = f(t_1, t_2)\sigma &\Rightarrow f(\alpha_1, \alpha_2)[S\sigma]\downarrow = \xi_i[S\sigma]\downarrow \\ &\Rightarrow f(\alpha_1, \alpha_2)[S'\sigma']\downarrow = \xi_i[S'\sigma']\downarrow \\ &\quad \text{car } S\sigma \sim S'\sigma \\ &\Rightarrow f(\alpha_1, \alpha_2)[S'\sigma']\downarrow = f(t'_1, t'_2)\sigma' \end{aligned}$$

D'où on a que $(S_2, C_2), (S'_2, C'_2)$ sont adéquats. □

Table des matières