# Modélisation formelle de protocoles RFID distance-bounding

Rapport de stage MPRI

Raphaëlle Crubillé, ENS Lyon
sous la direction de : Kostas Chatzikokolakis, LIX
et Myrto Arapinis, LFCS University of Edimburgh

22 août 2014

## Le contexte général

Les cartes à puce, permettant l'identification des objets et des personnes, sont maintenant omniprésentes dans tous les domaines de la société. La radio-identification (RFID) est une technologie qui permet aux puces de s'identifier à distance. Les tags RFID comprennent une antenne associée à la puce électronique qui leur permettent de recevoir et de répondre aux requêtes émises par le lecteur de carte. Le potentiel de la technologie RFID permet de l'appliquer non seulement à l'identification, mais aussi à des applications plus complexes comme le contrôle d'accès, la gestion des transports publics, le paiement... Dans ce but, des protocoles cryptographiques spécialement adaptés aux capacités des puces RFID ont été développés.

En pratique, le tag répond automatiquement à toutes les requêtes qu'il recoit. Le lecteur n'a donc pas de moyen de savoir si le porteur du tag est d'accord pour s'authentifier. Pour y remédier, on considère que le porteur donne implicitement son accord dès qu'il est placé assez près du lecteur. Donner au lecteur un moyen de déterminer si le tag est à une distance inférieure à une limite donnée est donc indispensable pour la sécurité des protocoles RFID.

## Le problème étudié

Les protocoles "distance-bounding" ont été introduits dans ce but par S.Brands et D.Chaum en 1993 [4] : un prouveur cherche à s'authentifier et à prouver qu'il est à proximité d'un vérificateur. Depuis, de nombreux protocoles ont été proposés [4], [8], [6]. Le point central de ces protocoles est usuellement une phase d'échange rapide, au cours de laquelle le vérificateur émet une requête, et mesure le temps qui s'écoule avant qu'il ne recoive une réponse.

A cause de la faible capacité de calcul des tags RFID, et du fait que durant la phase rapide, le temps de calcul doit être très inférieur au temps de propagation du signal, plusieurs de ces protocoles utilisent des primitives cryptographiques faibles, c'est-à-dire auxquelles on ne peut pas appliquer l'hypothèse de la cryptographie parfaite.

Nous avons cherché à exprimer les protocoles "distance-bounding"(DB) dans un modèle formel, dans le but de formaliser la notion d'attaque sur un protocole DB, et d'utiliser les techniques de preuves de protocoles développées pour les modèles formels. Une telle modélisation prenant en compte les aspects temporels et spatiaux des intéractions avait été dévellopée dans [2], mais conservait l'hypothèse de la cryptographie parfaites pour toutes les primitives. Dans [11], les auteurs construisait un modèle formel pour les protocoles DB, et prenaient en compte les probabilités que l'adversaire devine un terme à partir des informations qu'il a acquises. Cependant leur approche ne s'étend pas même à un exemple aussi simple que celui présenté à la section J de l'annexe.

## La contribution proposée

Nous avons proposé un modèle cryptographique hybride entre le modèle computationnel et le modèle symbolique. Comme dans le modèle symbolique, les messages sont modélisés par une algèbre de terme, et les agents sont modélisés par une algèbre de processus. L'égalité entre ces termes est probabiliste, et repose sur une distribution de probabilité sur les implémentations possibles des termes. L'adversaire est modélisé comme un processus opérant en temps polynomial, et on définit les propriétés de sécurité comme des bornes supérieures de la probabilité de succès d'un adversaire.

Plus précisément, nous avons défini une variante probabiliste du $\pi$-calcul appliqué, et deux sémantiques, labellées et non labellées, pour ce langage.

Nous avons ensuite étendu ce langage pour prendre en compte les phases rapides nécessaires au protocoles DB : au lieu de considérer directement l'aspect temporel, on considère qu'un processus peut choisir de déclencher une phase rapide, et que les règles de communications entre les processus sont modifiées pendant ces phases rapides : en particulier, la capacité des processus lointains à communiquer est restreinte.
Nous avons ensuite modélisé dans notre modèle les propriétés de sécurité pour des protocoles "distance-bounding", et les adversaires contre ces propriétés de sécurité. Nous avons exprimé deux protocoles de la littérature dans notre langage : le protocole proposé par Hancke et Kuhn dans [8], , et le protocole proposé par Brands et Chaum dans [4]. Nous avons exprimé dans notre modèle une attaque connue sur le protocole de Hancke et Kuhn, et nous avons prouvé une borne supérieure sur la probabilité qu'un adversaire polynomial placé loin du lecteur arrive à s'identifier illégitimement pour le protocole de Brands and Chaum.

## Les arguments en faveur de sa validité

On a uttilisé ce modèle pour formaliser et analyser deux protocoles de la littérature, et les résultats obtenus correspondent aux résultats du modèle computationnel. En particulier, notre modèle permet l'expression d'une attaque (donnée par Kim et al dans [9]) en temps linéaire et avec probabilité 1, sur le protocole proposé par Hancke Kuhn dans [8], qui n'est pas exprimable dans le modèle symbolique. De plus, on prouve une borne supérieure sur la probabilité de succès d'une attaque en temps polynomial sur le protocole proposé par Brands et Chaum dans [4], qui correspond (à un facteur polynomial près) à la borne supérieure donnée dans [4].

## Le bilan et les perspectives

Notre modèle hybride peut etre utilisé pour étudier d'autres propriétés de sécurité. En particulier, il serait intéressant d'exprimer des propriétés d'équivalence entre processus. On pourrait également étudier dans ce modèle des protocoles plus compliqués, et envisager une automatisation partielle des preuves. La question des liens entre ce modèle hybride et les modèles computationnels et symboliques devrait également etre explorée : en particulier, on pourrait étudier la question de l'existence d'un ensemble minimal de constructeurs tels qu'on puisse simuler le fonctionnement d'une machine de Turing probabiliste en temps polynomial, par un processus de ce langage évoluant en temps polynomial.

# 1 Introduction

Prouver qu'un protocole cryptographique vérifie une certaine propriété nécessite de modéliser ce protocole et cette propriété. Dans ce but, deux types de modèles communément utilisés ont été développés : le modèle computationnel et le modèle formel. Le modèle computationnel voit les messages comme des bitstrings, et les agents comme des machines de Turing probabilistes calculant en temps polynomial (PPT). Les propriétés de sécurité sont exprimées en terme de probabilité et de complexité des attaques. Le modèle symbolique repose sur l'hypothèse de la cryptographie parfaite : les primitives cryptographiques sont considérées comme des boites noires qui vérifient certaines propriétés. Dans ce modèle, les actions du protocole et les capacités de l'attaquant sont décrites par une théorie algébrique : c'est le modèle d'attaquant décrit par D.Yao et A.C Dolev dans [5]. En particulier, on considère les messages comme des termes générés par une algèbre. Les propriétés de sécurité sont modélisées de manière formelle. Les pouvoirs de l'adversaire sont alors très limités, mais il est plus facile de prouver la sécurité des protocoles, en particulier dans le cas où on considère des systèmes complexes avec plusieurs agents dans l'environnement. Différents outils d'automatisation (complète ou partielle) des preuves ont été développés, par exemple ProVerif sous la direction de B.Blanchet [3].

Néanmoins, il existe des protocoles dont les primitives cryptographiques ne peuvent pas être considérées comme des primitives parfaites. En particulier, des protocoles utilisés lorsque les agents ont de faibles capacités de calculs, ou bien devant effectuer le calcul dans un temps très réduit, peuvent faire intervenir des primitives faibles. C'est par exemple le cas des protocoles *distance-bounding*, dont le premier exemple a été présenté par S.Brands et D.Chaum en 1993 [4]. Ces protocoles sont conçus pour des tags RFID, et visent à certifier que la distance spatiale entre deux agents est inférieure à une valeur donnée : un agent envoie une requête et mesure le temps qui s'écoule avant que l'autre agent ne réponde à sa requête, et obtient une borne supérieure sur la distance qui les sépare. Pour cela, il faut que le temps pour calculer la réponse à partir de la requête soit très inférieur au temps de propagation du signal. Pendant les phases lentes du protocole, ces agents peuvent également utiliser des primitives cryptographiques "de haut niveau", c'est-à-dire qui peuvent être considérées comme parfaites (par exemple des fonctions de hashage, du chiffrement...)

Le protocole donné à la figure 1 (qui est une simplification du protocole donné par Y.Tu et S.Piramuthu dans [12]), est un protocole "distance bounding" : un lecteur et un tag RFID, qui partagent une clé secrète $s$, intéragissent entre eux, et le but du protocole est de borner la distance entre le lecteur et le tag. Pour cela, le lecteur mesure le temps que prends la phase rapide, c'est-à-dire le temps entre l'instant où il envoie $k$ et celui où il recoit $y$. La fonction $F$ utilisée dans ce protocole est un exemple de ce que l'on a appelé ci-dessus une primitive crypto faible. Le $i$-eme bit de $F(k, a, b)$ est calculé de la manière suivante :

$$(F(k,a,b))_i = \begin{cases} a_i \text{ si } k_i = 0 \\ b_i \text{ si } k_i = 1 \end{cases} \quad \text{avec } a_i,\ b_i,\ k_i \text{ respectivement les } i\text{-eme bit de } a,\ b,\ k$$

Le protocole se déroule de la manière suivante : les agents s'échangent des nonces, puis ils calculent tous les deux $a = h(N_R, N_T, s)$. Ensuite, le lecteur génère un name $k$, et déclenche son chronomètre au moment précis où il envoie $k$ au tag. Le tag répond alors par $F(y, a, a\,xor\,s)$, où $y$ est le message qu'il a recu. Le protocole réussit si le lecteur recoit une réponse avant une durée maximale fixée, et si cette réponse correspond bien à $F(k, a, a\,xor\,s)$

Il existe une attaque en temps linéaire (donnée par Kim et al dans [9]) qui permet à l'attaquant de connaître n'importe quel bit $i$ de la clé $s$ : l'adversaire relaie les nonces du début. Lorsque le lecteur émet le défi $k$, l'adversaire l'intercepte, et envoie au tag $\text{flip}_i(k)$, qui est le message obtenu en remplacant le $i$-eme bit de $k$ par sa négation. Ensuite, l'adversaire regarde si le protocole réussit ou non. Si le protocole réussit, cela signifie que $F(k, a, a\,xor\,s) = F(\text{flip}_i(k), a, a\,xor\,s)$, et donc que $s_i = 0$, sinon cela signifie que $s_i = 1$. Cette attaque est représentée à la figure 5 de l'annexe.

En revanche, il est impossible d'exprimer cette attaque dans un langage où l'égalité est donnée par une théorie équationnelle : en effet,les termes $F(k, a\,xor\,s, s)$ et $F(\text{flip}_i(k), a\,xor\,s, s)$ ne peuvent pas être symboliquement égaux.
Pour exprimer cette attaque, il faut que l'égalité soit modélisée de manière à ce que : avec probabilité $\frac{1}{2}$, ces deux termes sont égaux, avec probabilité $\frac{1}{2}$, ils sont différents.

L'approche suivie par D.Pavlovic et C.Meadows dans [11] pour analyser des protocoles de distance-bounding définit un modèle hybride entre le modèle symbolique et le modèle computationnel. Cependant, cette approche ne s'étend pas

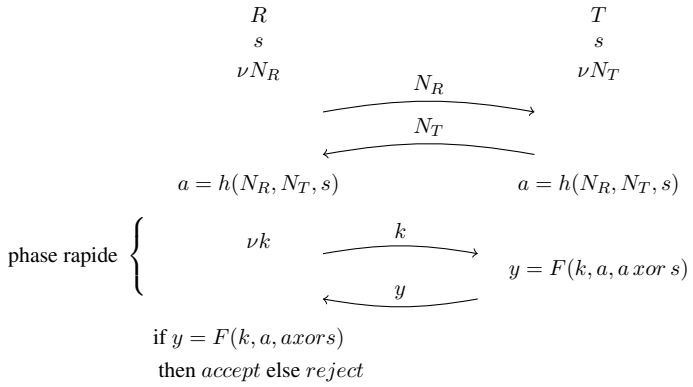



FIGURE 1 – Protocole Distance Bounding proposé par Tu et Pitramithu

même à un exemple aussi simple que celui présenté à la section J de l'annexe. Le but de ce travail est de reprendre cet objectif en utilisant des outils mathématiques plus adaptés, qui permettent d'exprimer à la fois des primitives cryptographiques fortes (que l'adversaire ne peut pas attaquer), et des primitives cryptographiques faibles. Pour cela, on définit un espace probabiliste correspondant aux interprétations computationnelles des termes, qui permet ensuite de définir une notion probabiliste d'égalité entre termes représentant des messages, et d'en déduire une sémantique probabiliste pour un langage de processus. On définit en fait une sémantique labellée et une sémantique non labellée, et on montre qu'on peut caractériser de manières équivalentes les propriétés de sécurité qui nous intéressent dans l'une ou l'autre sémantique. En utilisant cette équivalence, on exprime dans ce langage quelques attaques connues sur des protocoles distances-bounding, et on prouve une borne supérieure (cohérente avec les résultats existants) sur la probabilité de succès d'une attaque sur le protocole d'Hancke-Kuhn.

La structure de ce mémoire est la suivante : dans une première partie nous définissons les outils mathématiques pour définir l'égalité probabiliste entre termes, puis dans une deuxième partie nous introduisons le *$\pi$-calcul avec égalité probabiliste*, et dans une troisième partie nous utilisons une version modifiée de ce langage pour analyser des protocoles disatnces-boundings.

# 2 Syntaxe des termes

## 2.1 Spécification d'un langage de termes

On veut définir un langage de termes en spécifiant d'une part des constructeurs syntaxiques qui permettent de former des termes, et d'autre part leur interprétation computationnelle. Cette interprétation peut être déterministe (par exemple, on peut souhaiter spécifier le constructeur $xor$), mais elle peut aussi être probabiliste (par exemple, pour modéliser une fonction qui n'est pas manipulable par l'adversaire). On la représente donc comme une distribution sur l'ensemble des fonctions partielles de $(\{0,1\}^\star)^\alpha$ dans $\{0,1\}^\star$, avec $\alpha$ l'arité du constructeur considéré.

**Definition 1.** *Un langage de termes (implémentable) est un triplet $(\Sigma, \mathscr{N}, \mathscr{V})$, où :*
- *la signature $\Sigma$ est une liste finie d'éléments de la forme $(\mathbf{Cons}, \alpha, f)$, où le constructeur $\mathbf{Cons}$ est un mot sur un alphabet quelconque, $\alpha \in \mathbb{N}$ représente l'arité du constructeur, et $f : \mathbb{N}^\alpha \to \mathbb{N} \cup \{-1\}$ donne la longueur d'un terme en fonction des longueurs de ses arguments (la valeur -1 correspond au cas où ce constructeur n'est pas défini pour les longueurs demandées : par exemple le xor pour des arguments de longueur différentes.).*
- *$\mathscr{N}$ est un ensemble dénombrable, qui représente l'ensemble des names possibles, $\mathscr{V}$ est un ensemble dénombrable, qui représente l'ensemble des variables possibles, et on impose de plus que $\mathscr{N} \cap \mathscr{V} = \emptyset$.*

On fixe deux ensembles $\mathscr{N}$, $\mathscr{V}$ infinis, dénombrables et disjoints, et dans la suite, on pourra spécifier un langage uniquement par sa signature.

**Exemple 1.** *On peut définir par exemple les constructeurs suivants :*
- *les constantes* $\mathbf{un}$ *et* $\mathbf{zero}$*, d'arité 0, et de longueur* $((\cdot) \mapsto 1)$
- *le constructeur* $\mathbf{concat}$*, d'arité 2, et de fonction de longueur* $(p, q \mapsto p + q)$
- *les constructeurs* $\mathbf{enc}$ *et* $\mathbf{dec}$*, d'arité 2, et de fonction de longueur :* $((p, q) \mapsto q)$
- *le constructeur* $\mathbf{hash}$*, d'arité 2 et de fonction de longueur :* $\left(p \mapsto \frac{p}{2}\right)$
- *le constructeur* $\mathbf{xor}$*, d'arité 2 et de fonction de longueur :* $(p, q \mapsto p)$ *si* $p = q$, $-1$ *sinon*

*On appellera* ***Strings*** *le langage dont les constructeurs sont* $\mathbf{un}$*,* $\mathbf{zero}$ *et* $\mathbf{concat}$ *comme définis ci-dessus. On appellera* ***Enc*** *le langage dont les constructeurs sont* $\mathbf{enc}$*,* $\mathbf{dec}$ *et* $\mathbf{hash}$*.*

## 2.2 Grammaire des termes

**Definition 2.** *Soit* $\mathbf{\Lambda} = (\Sigma, \mathscr{N}, \mathscr{V})$ *un langage de termes. On note Terms*$(\mathbf{\Lambda})$*, l'ensemble des termes générés par la grammaire suivante :*

$$
\begin{array}{rll}
M & ::= & x^{\lambda} \qquad\qquad \text{pour tout } x \in \mathscr{V}, \lambda : \mathbb{N} \to \mathbb{N} \\
& \mid & n^{\lambda} \qquad\qquad \text{pour tout } n \in \mathscr{N}, \lambda : \mathbb{N} \to \mathbb{N} \\
& \mid & \mathbf{Cons}\,(M_1, ...M_\alpha) \qquad \text{pour tout } (\mathbf{Cons}, \alpha, f) \text{ un élément de } \Sigma
\end{array}
$$

**Exemple 2.** *On considère un langage qui contient les constructeurs* $\mathbf{concat}$ *et* $\mathbf{un}$ *définis à l'exemple 1. Alors si* $v \in \mathscr{V}$*, et* $n \in \mathscr{N}$*, le terme ci-dessous représente la concaténation de* $\mathbf{un}$*, de la variable* $v$ *et du name* $n$ *:*

$$\mathbf{concat}(\mathbf{un}, \mathbf{concat}(v^{(p \mapsto p+4)}, n^{(p \mapsto p)})) \text{ est un terme pour ce langage}$$

Dans la suite, on notera simplement $n$ et $x$ pour $n^{(p \mapsto p)}$ et $x^{(p \mapsto p)}$. On va maintenant définir une notion de longueur pour chaque terme : à chaque terme, on associe une longueur dans $\mathbb{N} \cup \{-1\}$ (qui est une fonction du paramètre de sécurité $l$). La valeur $-1$ correspond aux termes mal formés.

**Definition 3.** *Soit* $\mathbf{\Lambda}$ *un langage de termes. Soit* $l \in \mathbb{N}$ *qui représente le paramètre de sécurité. On définit une fonction (qui représente la longueur d'un terme)* $\mid . \mid_l$*: Terms*$(\mathbf{\Lambda}) \to \mathbb{N} \cup \{-1\}$*, qui est définie inductivement sur la structure des termes par :*

$$
\begin{array}{l}
\mid x^{\lambda} \mid_l = \lambda(l) \\
\mid n^{\lambda} \mid_l = \lambda(l) \\
\mid \mathbf{Cons}\,(M_1, .., M_\alpha) \mid_l = \begin{cases} f(\mid M_1 \mid_l, ..., \mid M_\alpha \mid_l) \text{ si } \mid M_1 \mid_l, ..., \mid M_\alpha \mid_l \neq -1 \\ -1 \text{ sinon} \end{cases} \qquad \text{pour } (\mathbf{Cons}, \alpha, f) \text{ un élément de } \mathbf{\Lambda},
\end{array}
$$

**Exemple 3.** *On considère un langage dont la signature contient les constructeurs* $\mathbf{xor}$*,* $\mathbf{un}$*, et* $\mathbf{concat}$ *définis à l'exemple 1. Alors pour tout paramètre de sécurité* $l \in \mathbb{N}$*, on a* $\mid \mathbf{concat}(\mathbf{un}, x^{(p \mapsto p+2)}) \mid_l = l+3$*, et* $\mid \mathbf{xor}(x^{(p \mapsto p)}, y^{(p \mapsto p+1)}) \mid_l = -1$ *(cela exprime le fait qu'on ne peut pas faire le xor bit à bit de deux termes de longueurs différentes)*

On dit qu'un terme $M$ est bien formé si, pour toute les valeurs du paramètre de sécurité $l$, $\mid M \mid_l \neq -1$. On appelle ground terme un terme bien formé qui ne contient pas de variables. On définit $\mathrm{LMax}(M)_l$ : l'indice maximal de $M$ comme la longueur du plus long sous-terme de $M$. De plus, on va définir une notion de complexité syntaxique : en effet, dans la suite, on cherchera à borner les pouvoirs de l'adversaire pour ne considérer que des attaques correspondant à un adversaire disposant d'un temps de calcul polynomial. La complexité syntaxique d'un terme correspond au nombre total de constructeurs dans ce terme, c'est-à-dire au nombre de fonctions qu'il est nécessaire d'appliquer pour calculer ce terme. On utilisera cette notion comme une caractérisation du temps de calcul demandé pour évaluer un terme.

**Exemple 4.** *On considère un langage dont la signature contient le constructeur* $\mathbf{exp}$ *d'arité 1, de fonction de longueur* $(n \mapsto 2 \times n)$ *(c'est-à-dire qui transforme un terme en un terme deux fois plus long), et le constructeur* $\mathbf{red}$ *d'arité 1, de fonction de longueur* $\left(n \mapsto \frac{n}{2}\right)$ *(qui compresse un terme d'un facteur 2). On considère le terms* $M_e \stackrel{def}{=} \mathbf{red}^{10}(\mathbf{exp}^{10}(x^{(p \mapsto p)}))$*. On a alors* $\mid M_e \mid_l = l$*, et LMax*$(M_e)_l = 2^{10} \cdot l$*. La complexité syntaxique de* $M_e$ *est 20.*

# 3 Interprétation des termes du langage

On veut maintenant spécifier une interprétation computationnelle pour ce langage, c'est-à-dire un moyen d'associer de manière probabiliste une chaîne de charactères sur l'alphabet $\{0,1\}$ à chaque terme du langage. On va faire cela en trois temps : on définit d'abord (à la section 3.1) une interprétation computationnelle des constructeurs, puis on définira (à la section 3.2) une interprétation computationnelle des names. On les combinera ensuite (à la section 3.3) pour obtenir une interprétation computationnelle des termes.
Puisque ces interprétations sont probabilistes, on doit les spécifier comme des mesures de probabilité sur une $\sigma$-algèbre (les définitions d'une $\sigma$-algèbre et d'une mesure de probabilité sont rappelées dans l'annexe C).

## 3.1 Interprétation des constructeurs du langage

On va maintenant définir une $\sigma$-algèbre dont les éléments sont des ensembles d'implémentations possibles pour les constructeurs du langage. On introduit la notation suivante : pour $E$ et $F$ deux ensembles quelconques, $E \rightharpoonup F$ est l'ensemble des fonctions partielles de $E$ dans $F$.

**Definition 4.** *Soit* $\mathbf{Cons}$ *un mot sur un alphabet quelconque,* $\alpha \in \mathbb{N}$, $f : \mathbb{N}^\alpha \to \mathbb{N} \cup \{-1\}$. *Une implémentation d'un constructeur* $(\mathbf{Cons}, \alpha, f)$ *est un élément de l'ensemble :*

$$\mathit{ImplCons}(\mathbf{Cons}, \alpha, f) = \{g : (\{0,1\}^\star)^\alpha \rightharpoonup \{0,1\}^\star \mid \forall x = (x_1, ..., x_\alpha),\ |g(x)| = f(|x_1|, ..., |x_\alpha|)\},$$

*où pour une bitstring* $z$, $|z|$ *représente la longeur de* $z$, *et on note* $|z| = -1$ *si* $z$ *n'est pas définie.*

**Exemple 5.** *On considère un constructeur d'arité* 2, *de fontion de longueur* $f = (p, q \mapsto p$ *si* $p = q$, $-1$ *sinon* $)$. *Alors :*

- *la fonction* $\big((x,y) \in (\{0,1\}^\star)^2 \mapsto x$ *xor* $y$ *si* $|x| = |y|$, *n'est pas défini sinon* $\big)$ *est une implémentation de ce constructeur : A deux mots* $x$ *et* $y$ *sur l'alphabet* $\{0,1\}$, *on associe* $(x$ *xor* $y)$ *si* $x$ *et* $y$ *sont deux mots de la même longueur, et rien sinon (ce qui est possible puisqu'il s'agit d'une fonction partielle).*
- $\big((x,y) \in (\{0,1\}^\star)^2 \mapsto x$ *si* $|x| = |y|$, *n'est pas défini sinon* $\big)$ *est une autre implémentation possible de ce constructeur.*
- *en revanche, la fonction* $\big((x,y) \in (\{0,1\}^\star)^2 \mapsto x\big)$ *n'est pas une implémentation de ce constructeur : en effet, cette fonction est définie sur des mots* $x$ *et* $y$ *de longueur différentes, ce qui est en contradiction avac la définition de f.*

**Definition 5.** *Soit* $\mathbf{\Lambda}$ *un langage de signature* $\Sigma$. *On définit* l'ensemble des implémentations possibles d'une signature $\mathbf{\Lambda}$ *par :*

$$\mathit{ImplSig}(\mathbf{\Lambda}) = \prod_{(\mathbf{Cons}, \alpha, f) \in \Sigma} \mathit{ImplCons}(\mathbf{Cons}, \alpha, f)$$

**Exemple 6.** *On s'intéresse au langage* ***Strings*** *défini à l'exemple 1. L'exemple ci-dessous illustre le fait que le même langage admet plusieurs implémentations différentes possibles de sa signature. Ainsi, l'implémentation ci-dessous est l'implémentation "naturelle" du langage* ***Strings*** *: elle associe* 1 *au contructeur* $\mathbf{un}$, 0 *au constructeur* $\mathbf{zero}$, *et associe au constructeur* $\mathbf{concat}$ *la fonction qui effectue la concaténation de deux chaînes de caractères.*

$$\eta^{nat}_{\mathbf{Strings}} \stackrel{def}{=} \left((\cdot \mapsto 1), (\cdot \mapsto 0), (x, y \mapsto x \cdot y)\right) \in \mathit{ImplSig}(\mathbf{Strings}).$$

*Mais il existe d'autres implémentations possibles de cette signature : on peut associer* 1 *aux constructeur* $\mathbf{un}$ *et* $\mathbf{zero}$, *et associer au constructeur* $\mathbf{concat}$ *la fonction qui produit une bitstring constituée unniquement de* 0 *(et de longueur adéquate) :*

$$\left((\cdot \mapsto 1), (\cdot \mapsto 1), ((x,y) \mapsto (0)^{|x|+|y|})\right) \in \mathit{ImplSig}(\mathbf{Strings}).$$

On remarque que l'ensemble des implémentations possibles de la signature est non dénombrable. Mais on n'est en réalité pas intéressé par le fait d'attribuer une probabilité à tous les ensembles d'implémentations, mais seulement à ceux qui sont caractérisés par des contraintes sur les restrictions de ces implémentations aux bitstrings d'une certaine

taille. On va formaliser cette idée de la manière suivante : on définit $AlgSig_{\mathbf{\Lambda}}$ comme la $\sigma$-algèbre construite à partir des implémentations partielles des constructeurs, c'est-à-dire aux implémentations qui sont définies seulement sur les bistrings de longueur inférieure à un entier $m$ donné, et une interprétation de la signature est une mesure de probabilité sur cette $\sigma$-algèbre.

**Definition 6.** *Soit $\mathbf{\Lambda} = (\Sigma, \mathscr{N}, \mathscr{V})$ un langage de termes (implémentable).*

- *Pour un constructeur $(\mathbf{Cons}, \alpha, f)$ donné, et $m \in \mathbb{N}$, $ImplCons_m(\mathbf{Cons}, \alpha, f)$ est l'ensemble des implémentations partielles de la signature pour $m$, c'est-à-dire des fonctions partielles : $(\{0,1\}^{\leq m})^{\alpha} \rightharpoonup \{0,1\}^{\star}$, qui renvoient des bitstrings de longeurs adéquates. On définit l'ensemble des implémentations partielles de la signature : $ImplSig_m(\mathbf{\Lambda})$ comme le produit des ensembles d'implémentations partielles de chacun des constructeurs de $\mathbf{\Lambda}$.*
- *Pour $\eta$ une implémentation partielle de la signature pour $m$, on définit $\uparrow^{sig}(\eta)_m^{\mathbf{\Lambda}}$ l'ensemble des* implémentations dirigées par *$\eta$ comme l'ensemble des implémentations qui se comportent comme $\eta$ sur des arguments de longeur inférieure à $m$.*

**Exemple 7.** *Pour chaque $m \in \mathbb{N}$, on peut définir une implémentation partielle "naturelle" du langage **Strings**, qui correpond à la restriction à $(\{0,1\}^m)$ de l'implémentatation "naturelle" définie à l'exemple 6 : On définit*

$$\eta_m^{nat} \stackrel{def}{=} \left( ((\cdot) \to 1, (\cdot) \to 0, (x,y) \in (\{0,1\}^{\leq m})^2 \to (0)^{|x|+|y|}) \right).$$

*On s'intéresse ensuite à l'ensemble des implémentations dirigées par $\eta_m^{nat}$. On peut voir en particulier que : $((\cdot) \to 1, (\cdot) \to 0$ $, ((x,y) \to (0)^{|x|+|y|}$ si $|x| \leq m+5, x.y$ sinon $)) \in \uparrow^{sig}(\eta_m^{nat})_m^{\mathbf{Strings}}$. En effet cette fonction se comporte comme $\eta_{\mathbf{Strings}}^{nat}$ pour $x, y$ de longueur inférieure à $m$.*

On peut maintenant définir $AlgSig_{\mathbf{\Lambda}}$ la $\sigma$-algèbre correspondant à l'interprétation de la signature de $\mathbf{\Lambda}$ : c'est la $\sigma$-algèbre sur $\mathrm{ImplSig}(\mathbf{\Lambda})$ dont les éléments générateurs sont les $\uparrow^{\mathrm{sig}}(\eta)_m^{\mathbf{\Lambda}}$. Ainsi, tout élément de $AlgSig_{\mathbf{\Lambda}}$ est construit à partir d'unions dénombrables, d'intersections finies et de passages au complémentaire à partir de ces éléments.

**Definition 7.** *Soit $\mathbf{\Lambda}$ un langage de termes (implémentable). On définit $AlgSig_{\mathbf{\Lambda}}$ comme étant la plus petite $\sigma$-algèbre qui contient les $\uparrow^{sig}(\eta)_m^{\mathbf{\Lambda}}$ pour tout $m \in \mathbb{N}$ et $\eta \in ImplSig_m(\mathbf{\Lambda})$.*

On peut voir la différence entre $AlgSig_{\mathbf{\Lambda}}$ et $\mathcal{P}(\mathrm{ImplSig}(\mathbf{\Lambda}))$ (l'ensemble des parties de $\mathrm{ImplSig}(\mathbf{\Lambda})$, qui est aussi une $\sigma$-algèbre), en observant que : pour tout langage de termes $\mathbf{\Lambda}$, quelque soit $\eta \in \mathrm{ImplSig}(\mathbf{\Lambda})$, le singleton $\{\eta\} \notin AlgSig_{\mathbf{\Lambda}}$, alors que $\{\eta\} \in \mathcal{P}(\mathrm{ImplSig}(\mathbf{\Lambda}))$. Définir ainsi la $\sigma$-algèbre que l'on considère permet ensuite de pouvoir définir une mesure de probabilité sur cette $\sigma$-algèbre en la définissant seulement sur les éléments générateurs de $AlgSig_{\mathbf{\Lambda}}$.

**Definition 8.** *Soit $\mathbf{\Lambda}$ un langage de termes (implémentable). Une interpretation de la signature de $\mathbf{\Lambda}$ est une mesure de probabilité $\mathscr{D}_{sig}$ sur $AlgSig_{\mathbf{\Lambda}}$ (i.e $\mathscr{D}_{sig} : AlgSig_{\mathbf{\Lambda}} \to [0,1]$).*

On remarque que pour définir une mesure de probabilité $\mathscr{D}$ sur $AlgSig_{\mathbf{\Lambda}}$, il est suffisant de définir, pour tout $m$, pour tout $\eta$, $\mathscr{D}(\uparrow^{\mathrm{sig}}(\eta)_m^{\mathbf{\Lambda}})$ (la preuve est en annexe D.2).

**Exemple 8.** *On peut définir l'interprétation "naturelle"(et déterministe) des constructeurs du langage **Strings** en donnant une probabilité $1$ à l'implémentation naturelle $\eta_{\mathbf{Strings}}^{nat}$. Pour tout $m$, pour tout $\eta \in ImplSig(\mathbf{Strings})$ :*

$$\mathscr{D}(\uparrow^{sig}(\eta)_m^{\mathbf{Strings}}) = \begin{cases} 1 \text{ si } \eta = \eta_m^{nat} & \text{avec } \eta_m^{nat} \text{ l'implémentation partielle définie à l'exemple 7.} \\ 0 \text{ sinon} \end{cases}$$

**Exemple 9.** *Supposons que l'on veuille disposer d'une fonction de compression par un facteur 2, qui vérifie le modèle de l'oracle aléatoire : pour chaque appel à cette fonction, la sortie est une chaine aléatoire (de la bonne longueur) si c'est la première fois que cette fonction est appelée sur cette entrée, et le résultat précédent si la fonction a déjà été appelée sur cette entrée. On peut exprimer cela dans ce modèle en disant qu'il s'agit d'une fonction dont la représentation computationnelle (pour une entrée de longueur $m$) est la distribution uniforme sur l'ensemble des fonctions $\{0,1\}^m \to \{0,1\}^{\frac{m}{2}}$.*

*On veut disposer d'un langage* ***Oracle****, dont la signature est égale à :* $\Sigma^{\textbf{\textit{Oracle}}} = (h, 1, (p \mapsto \frac{p}{2}))$*. et on définit sa représentation computationnelle par : pour tout* $\eta \in$ *ImplSig*$_m$ (***Oracle***) *:*

$$\mathscr{D}(\uparrow^{sig}(\eta)_m^{\textbf{\textit{Oracle}}}) = \frac{1}{\#\textit{ImplSig}_m(\textbf{\textit{Oracle}})}$$

**Exemple 10.** *On considère le langage* ***Enc*** *défini à l'exemple 1. On va définir une représentation computationnelle des constructeurs pour laquelle* $\mathbf{hash}$ *vérifie le modèle de l'oracle aléatoire, et pour laquelle* $\mathbf{enc}$ *et* $\mathbf{dec}$ *sont choisi uniformément dans l'espace des fonctions qui vérifient l'équation caractéristique des primitives de chiffrement :* $\forall x_k, x_m \in \{0,1\}^l$*, dec*$(x_k, enc(x_k, x_m) = x_m$*. Pour tout* $m \in \mathbb{N}$*, on considère l'ensemble*

$$E_m = \{(\eta_{enc}, \eta_{dec}, \eta_{hash}) \in \textit{ImplSig}_m(\textbf{\textit{Enc}}) \mid \forall p, q \textit{ tel que } p, q \leq m, \forall x, y \in \{0,1\}^p \times \{0,1\}^q,\ \eta_{dec}(x, \eta_{enc}(x, y))) = y\},$$

*et on définit la représentation computationnelle des constructeurs par : pour tout* $\eta \in$ *ImplSig*$_m$ (***Enc***)

$$\mathscr{D}(\uparrow^{sig}(\eta)_m^{\textbf{\textit{Enc}}}) = \begin{cases} \frac{1}{\#E_m} \textit{ si } \eta \in E_m \\ 0 \textit{ sinon} \end{cases}$$

## 3.2 Interprétation des names

De manière analogue, pour chaque paramètre de sécurité $l$, on définit l'ensemble des implémentations possible des names comme l'ensemble des fonctions qui à un name et à sa fonction de longueur associent une bitstring de longueur adéquate :

$$\text{ImplNames}(\mathbf{\Lambda}, l) \stackrel{\text{def}}{=} \{\Theta : \mathscr{N} \times \{\text{f} : \mathbb{N} \to \mathbb{N}\} \to \{0,1\}^\star \mid |\Theta(n, \text{f})| = \text{f}(l)\}$$

On peut ensuite définir une $\sigma$-algèbre $AlgNames_l^{\mathbf{\Lambda}}$ sur l'ensemble ImplNames$(\mathbf{\Lambda}, l)$, qui correspond aux implémentations possibles des names (la construction se trouve à l'annexe D.3 ). Comme précédemment, une interprétation des termes est une mesure de probabilité sur cette $\sigma$-algèbre.

**Definition 9.** *Soit* $\mathbf{\Lambda}$ *un langage de termes (implémentable),* $l \in \mathbb{N}$ *un paramètre de sécurité. Une interpretation des names de* $\mathbf{\Lambda}$ *pour le paramètre de sécurité* $l$ *est une mesure de probabilité* $\mathscr{D}_{names}$ *sur* $AlgNames_l^{\mathbf{\Lambda}}$ *(i.e* $\mathscr{D}_{names} : AlgNames_l^{\mathbf{\Lambda}} \to [0,1]$*)*

On peut définir une interprétation des names $\mathscr{D}_{namesl}^{\text{unif}}$ qui correspond à la distribution uniforme, c'est-à-dire telle que l'implémentation d'un name est indépendante des implémentations des autres names, et toutes les bitstrings de longueur adéquate ont la même probabilité de représenter un name (la construction de cette interprétation se trouve en annexe D.3). En particulier, pour tout name $n$, pour tout f : $\mathbb{N} \to \mathbb{N}$, pour toute bitstring $m$ de longueur f$(l)$, on a :

$$\mathscr{D}_{namesl}^{\text{unif}}(\{\Theta \mid \Theta(n, \text{f}) = m\}) = \frac{1}{2^{\text{f}(l)}}$$

Dans la suite, on choisira systématiquement la représentation uniforme comme représentation des names.

Pour pouvoir définir plus loin des processus dont la sémantique est stable par $\alpha$-conversion, on ne peut donner que des interprétations qui sont invariante par substitution des names.

## 3.3 Interprétation computationnelle des termes du langage :

On va maintenant définir la notion d'implémentation : une implémentation correspond à une fonction : Terms$(\mathbf{\Lambda}) \to \{0,1\}^\star$, pour chaque valeur du paramètre de sécurité $l$ :

**Definition 10.** *Soit* $\mathbf{\Lambda}$ *un langage, soit* $l \in \mathbb{N}$

- *On définit* l'ensemble des implémentations des termes pour $l$ *: ImplLan*$(l, \mathbf{\Lambda})$ *= ImplSig* $(\mathbf{\Lambda}) \times$ *ImplNames*$(\mathbf{\Lambda}, l)$
- *On définit la* $\sigma$*-algèbre* $AlgLan_l^{\mathbf{\Lambda}}$ *sur l'ensemble ImplLan*$(l, \mathbf{\Lambda})$ *comme l'algèbre produit de* $AlgSig_{\mathbf{\Lambda}}$ *et* $AlgNames_l^{\mathbf{\Lambda}}$*, c'est-à-dire l'algèbre engendrée par les parties de la forme* $A \times B$*, avec* $A \in AlgSig_{\mathbf{\Lambda}}$ *et* $B \in AlgNames_l^{\mathbf{\Lambda}}$*.*

**Exemple 11.** *On considère le langage **Strings**. Alors l'implementation ci-dessous correspond à une interprétation naturelle des constructeurs, et une interprétatation des names comme étant constitués uniquement de* $0$ *:*

$$\eta = \left(\left((\cdot \mapsto 1), (\cdot \mapsto 0), (x, y \mapsto x \cdot y)\right), \left((n,f) \mapsto 0^{f(l)}\right)\right) \in \mathit{ImplLan}(l, \mathbf{\Lambda})$$

**Definition 11.** *Soit* $\mathbf{\Lambda}$ *un langage. Une interprétation computationnelle des termes est une suite* $\mathscr{M} = (\mathscr{M}_l)_{l\in\mathbb{N}}$*, telle que pour tout paramètre de sécurité* $l \in \mathbb{N}$*,* $\mathscr{M}_l$ *est une mesure de probabilité sur la* $\sigma$*-algèbre* $AlgLan_l^{\mathbf{\Lambda}}$*.*

Soit $\mathbf{\Lambda}$ un langage, et $\mathscr{D}_{sig}$ une interprétation des constructeurs de $\mathbf{\Lambda}$, et pour tout paramètre de sécurité $l$, $\mathscr{D}_{names}^l$ une interprétation des names de $\mathbf{\Lambda}$ pour le paramètre de sécurité $l$. On veut maintenant en déduire une interprétation des termes, c'est-à-dire une mesure de probabilité sur la $\sigma$-algèbre $AlgLan_l^{\mathbf{\Lambda}}$ pour chaque paramètre de sécurité $l \in \mathbb{N}$.

**Definition 12.** *Soit* $\mathbf{\Lambda}$ *un langage, soit* $\mathscr{D}_{sig}$ *une interprétation des constructeurs de* $\mathbf{\Lambda}$*, et soit* $\mathscr{D}_{names} = (\mathscr{D}_{names}{}^l)_{l\in\mathbb{N}}$ *une suite tel que pour chaque* $l \in \mathbb{N}$*,* $\mathscr{D}_{names}^l$ *est une interprétation des names de* $\mathbf{\Lambda}$ *pour le paramètre de sécurité* $l$*. Alors on définit une interprétation computationnelle des termes* $\mathscr{M}(\mathscr{D}_{sig}, \mathscr{D}_{names})$ *par :* $\forall l \in \mathbb{N}$*,* $\mathscr{M}(\mathscr{D}_{sig}, \mathscr{D}_{names})_l$ *est la mesure produit de* $\mathscr{D}_{sig}$ *et* $\mathscr{D}_{names}^l$

A une implémentation des termes de $\mathbf{\Lambda}$, on peut associer une fonction qui a tout terme du langage $\mathbf{\Lambda}$ associe un mot sur l'alphabet $\{0, 1\}$.

**Definition 13.** *On considère un langage* $\mathbf{\Lambda}$*, et* $\eta$ *une implémentation des termes de ce langage pour le paramètre de sécurité* $l$*. On peut décomposer* $\eta$ *en une implémentation des names* $\eta_{names}$*, et une implémentation* $\eta_{\mathbf{Cons}}$ *pour chaque constructeur appartenant à la signature de* $\mathbf{\Lambda}$*. On définit inductivement* $\eta^\star$ *:* $\mathit{Terms}(\mathbf{\Lambda}) \to \{0,1\}^\star$ *par :*

$$\eta^\star(n^f) = \eta_{names}(n,f)$$
$$\eta^\star(\mathbf{Cons}(M_1, ..., M_\alpha)) = \eta_{\mathbf{Cons}}(\eta^\star(M_1), \cdots, \eta^\star(M_\alpha)))$$

**Exemple 12.** *On considère le langage **Strings**. Soit* $l \in \mathbb{N}$*. On considère l'implémentation* $\eta$ *formée de l'implémentation naturelle* $\eta^{nat}$ *sur les constructeurs, et de l'implémentation uniforme sur les names pour le paramètre* $l$*. On a alors :* $\eta^\star(\mathbf{concat}(\mathbf{un}, \mathbf{zero})) = 10$

## 3.4 Egalité

On cherche maintenant à définir la probabilité que, à un certain stade de l'exécution, deux termes soient égaux. En effet, on veut pouvoir exprimer que le processus $\text{if } (M = N) \text{ then } P \text{ else } Q$ va dans la branche $P$ avec une certaine probabilité $p$, et dans la branche $Q$ avec probabilité $1 - p$. De plus, si on a choisi la branche $P$, et que l'on refait un test d'égalité, la probabilité de succès doit prendre en compte le fait que $M = N$. Ainsi, la probabilité que deux termes soient égaux doit en fait dépendre de tous les tests effectués auparavant. Pour cette raison, dans cette section on va introduire des contraintes, qui représentent le résultat des tests déjà effectués, et on va définir la probabilité que deux termes $M$ et $N$ sont *égaux sachant qu'une certaine contrainte* $\mathcal{C}$ *est vérifiée*.

### 3.4.1 Contraintes

Une contrainte est une conjonction de formules de la forme $(M = N)$ ou $\text{not}\,(M = N)$. Elle exprime les égalités et non-égalités qui doivent être vérifiées à ce point de l'exécution.

**Definition 14.** *Soit* $\mathbf{\Lambda}$ *un langage. L'ensemble* $Constraints\,(\mathbf{\Lambda})$ *des contraintes définies sur ce langage est donné par la grammaire suivante :*

$$\mathcal{C} \;::=\; \top \;\Big|\; (M = N) \wedge \mathcal{C} \;\Big|\; \mathit{not}\,(M = N) \wedge \mathcal{C}$$

*où* $M$ *et* $N$ *sont des ground termes de* $\mathbf{\Lambda}$*.*

**Exemple 13.** *On considère un langage dont la signature contient les constructeurs* $\mathbf{un}$, $\mathbf{zero}$, $\mathbf{concat}$ *(par exemple, le langage* ***Strings****). Alors la contrainte* $\mathcal{C}_1 \stackrel{def}{=} (\mathbf{un} = \mathbf{zero} \wedge \mathbf{concat}(\mathbf{un}, \mathbf{zero}) = \mathbf{zero})$*, et également la contrainte* $\mathcal{C}_2 \stackrel{def}{=}$ *not* $\left(\mathbf{concat}(\mathbf{un}, n^{(p \mapsto p)}) = m^{(p \mapsto p+1)}\right)$ *sont des contraintes définies sur ce langage.*

Etant donnée une contrainte, on s'intéresse à l'ensemble des implémentations du langage telles que cette contrainte soit vraie.

**Definition 15.** *Soit* $\mathbf{\Lambda}$ *un langage,* $l \in \mathbb{N}$*. Soit* $\mathcal{C}$ *une contrainte sur ce langage. On définit* $[\![\mathcal{C}]\!]_l$ *l'ensemble inclus dans* $\mathit{ImplLan}(l, \mathbf{\Lambda})$ *défini par induction sur la structure de* $\mathcal{C}$ *de la manière suivante :*

- *Si* $\mathcal{C} = \top$, $[\![\mathcal{C}]\!]_l = \mathit{ImplLan}(l, \mathbf{\Lambda})$
- *Si* $\mathcal{C} = (M = N) \wedge \mathcal{C}_1$*, alors* $[\![\mathcal{C}]\!]_l = \{\eta \in [\![\mathcal{C}_1]\!]_l$ *tel que* $\eta^\star(M) = \eta^\star(N)\}$
- *Si* $\mathcal{C} =$ *not* $((M = N)) \wedge \mathcal{C}_1$*, alors* $[\![\mathcal{C}]\!]_l = \{\eta \in [\![\mathcal{C}_1]\!]_l$ *tel que* $\eta^\star(M) \neq \eta^\star(N)\}$

**Exemple 14.** *On s'intéresse aux implémentations du langage* ***Strings*** *qui vérifient les contraintes de l'exemple 13 :*

- $[\![\mathcal{C}_1]\!]_l = \emptyset$ *: en effet, le terme* $\mathbf{concat}(\mathbf{un}, \mathbf{zero})$ *et le terme zero ont des longueur différentes : il n'existe aucune implémentation qui les rende égaux.*
- $[\![\mathcal{C}_2]\!]_l = \{\eta \in \mathit{ImplLan}(l,$ ***Strings***$)$ *tel que* $\eta^\star(concat(un, n^{(p \mapsto p)}) \neq \eta^\star(m^{(p \mapsto p+1)})\}$

Le lemme ci-dessous indique que, pour une contrainte sur le langage $\mathbf{\Lambda}$, et pour $l$ un paramètre de sécurité, l'ensemble des interprétations pour ce paramètre de sécurité qui vérifient la contrainte est un élément de la $\sigma$-algèbre $AlgLan_l^{\mathbf{\Lambda}}$ : cela signifie que si on a une interprétation computationnelle des termes $\mathscr{M}$ qui correspond à une mesure de probabilité sur $AlgLan_l^l$, on peut associer une probabilité au fait que cette contrainte soit vérifiée.

**Lemme 1.** *Soit* $\mathbf{\Lambda}$ *un langage,* $l$ *un paramètre de sécurité. Pour toute contrainte* $\mathcal{C}$ *sur* $\mathbf{\Lambda}$*, on a :* $[\![\mathcal{C}]\!]_l \in AlgLan_l^{\mathbf{\Lambda}}$.

**Exemple 15.** *On considère le langage* ***Strings****, et* $\mathscr{D}_{sig}^{nat}$ *l'interprétation naturelle des constructeurs de* ***Strings*** *definie à l'exemple 8, et la suite* $\mathscr{D}_{names}^{unif} = (\mathscr{D}_{namesl}^{unif})_{l \in \mathbb{N}}$ *l'interprétation uniforme des names Alors*

$$\mathscr{M}(\mathscr{D}_{sig}^{nat}, \mathscr{D}_{names}^{unif})_l \left( [\![(\mathbf{concat}(\mathbf{zero}, n^{(p \mapsto p)}) = \mathbf{concat}(\mathbf{un}, m^{(p \mapsto p)})]\!]_l \right) = 0$$

### 3.4.2 Cas particulier d'une fonction modélisant l'oracle aléatoire

Pour une primitive cryptographique, donner l'interprétation correpondant à la distribution uniforme sur toutes les fonctions qui vérifient les conditions demandées (qui correspondent aux équations données en cryptographie symbolique, par exemple $\mathbf{dec}(m_k, \mathbf{enc}(m_k, m_e))) = m_e$), revient à la considérer comme parfaite (à une probabilité négligeable près) : en effet, rendre le choix de la fonction aléatoire dans l'ensemble des fonctions possibles empêche l'adversaire de pouvoir briser cette primitive. On va le montrer pour le cas particulier de la fonction de hashage.

On considère un langage dont la signature $\Sigma$ contient les constructeurs $\mathbf{un}$, $\mathbf{zero}$, $\mathbf{concat}$, et le constructeur $\mathbf{hash}$ de fonction de longueur f. On veut considérer une interprétation de la signature telle que l'interprétation de $\mathbf{hash}$ est indépendante de l'interprétation des autres constructeurs, et qu'elle modélise l'oracle aléatoire, et de même que $\mathbf{un}$, $\mathbf{zero}$ et $\mathbf{concat}$ aient leur interprétation naturelle. On exprime cela par : soit $\mathscr{M} = \mathscr{M}(\mathscr{D}_{sig}, \mathscr{D}_{names}^{\text{unif}})$ une interprétation de ce langage telle que : $\mathscr{D}_{sig} = \mathscr{D}_{\mathbf{hash}} \times \mathscr{D}_{\mathbf{Strings}} \times \mathscr{D}_{\Sigma \setminus \mathbf{Strings}, \mathbf{hash}}$, avec :

- $\mathscr{D}_{\mathbf{Strings}}$ est l'interprétation naturelle (définie à l'exemple 8 ) pour les constructeurs $\mathbf{un}$, $\mathbf{zero}$, $\mathbf{concat}$.
- $\mathscr{D}_{\mathbf{hash}}$ est l'interprétation correspondant à l'oracle aléatoire (définie à l'exemple 9) pour le constructeur $\mathbf{hash}$, avec f sa fonction de longueur.

Le constructeur $\mathbf{hash}$ ainsi défini peut être vu comme une primitive cryptographique parfaite, dans le sens suivant : l'adversaire ne peut pas en déduire plus d'informations qu'il n'en déduirait d'une chaîne de caractère quelconque. Le lemme ci-dessous (les preuves de cette section sont en annexe D.4) explicite cette idée : on peut remplacer tout sous-terme de la forme $\mathbf{hash}(M)$ par un name , c'est-à-dire que pour n'importe quelle contrainte, la probabilité que cette contrainte soit vérifiée peut être exprimée comme une somme de probabilité de contraintes qui ne mentionnent pas $\mathbf{hash}$.

**Lemme 2.** *Soit $\mathcal{C}$ une contrainte, soit $M_1, ..., M_r$ les termes tels que $\mathbf{hash}(M_i)$ est un sous-terme d'un terme apparaissant dans $\mathcal{C}$. Alors, pour $n_1, .., n_r$ des fresh names :*

$$\mathscr{M}(\llbracket\mathcal{C}\rrbracket_l) \leq \sum_{I_1,..,I_k \text{ partition de } \{1,...,r\}} \mathscr{M}(\llbracket\mathcal{C}\,\{\forall j, n_{\bar{j}}/\mathbf{hash}(M_j)\}$$
$$\wedge\, M_{i_1}\{\forall j, n_{\bar{j}}/\mathbf{hash}(M_j)\} = M_{i_2}\{\forall j, n_{\bar{j}}/\mathbf{hash}(M_j)\} \forall i_1,\, i_2 \text{ tel que } \bar{i}_1 = \bar{i}_2$$
$$\wedge\, not\left(M_{i_1}\{\forall j, n_{\bar{j}}/\mathbf{hash}(M_j)\} = M_{i_2}\{\forall j, n_{\bar{j}}/\mathbf{hash}(M_j)\}\right)\, \forall i_1,\, i_2 \text{ tel que } \bar{i}_1 \neq \bar{i}_2\rrbracket_l)$$

*où, si $I_1, ..., I_k$ est une partition de $\{1, ..., r\}$, on note $\bar{i} = j$ pour $i \in I_j$.*

**Exemple 16.** *Soit $s_1$, $s_2$ deux names. On considère la contrainte $\mathbf{hash}(s_1) = \mathbf{hash}(s_2)$. En appliquant le lemme, on obtient :*

$$\mathscr{M}(\llbracket\mathbf{hash}(s_1) = \mathbf{hash}(s_2)\rrbracket_l) \leq \mathscr{M}\left(\llbracket n = m \wedge not\,(s_1 = s_2)\rrbracket_l\right) + \mathscr{M}\left(\llbracket n = n \wedge s_1 = s_2\rrbracket_l\right)$$

La fonction ainsi définie a bien les caractéristiques demandées à un fonction de hashage computationnelle. En particulier, le lemme ci-dessous entraîne la propriété dite de "préimage résistance", qui est demandée à une fonction de hashage dans le modèle computationnel : un adversaire polynomial ne peut pas, avec une probabilité non négligeable, trouver une bitstring dont l'image par $\mathbf{hash}$ est un élément fixé à l'avance.

**Lemme 3.** *Soit $(\mathcal{C}_i)_{i\in I}$ une famille de contraintes générées par un adversaire, et telles que pour tout $i$, la complexité syntaxique de $\mathcal{C}_i$ est inférieure à $r$, et soit $(M_i)_{i\in I}$ une famille de terme de longueur $p$ telle que pour tout $i$, la complexité syntaxique de $M_i$ est inférieure à $r$.*
*Soit $l$ un paramètre de sécurité. Soit $A \subseteq \{0,1\}^{f(|M|_l)}$. Alors il existe un polynome $P$ tel que :*

$$\sum_{1\leq i\leq n} \llbracket\mathbf{hash}(M_i) \in A \wedge \mathcal{C}_i\rrbracket_l \leq \#(A) \times \frac{P(r)}{2^{fp}}$$

En particulier, le lemme ci-dessous entraîne entre autre la propriété dite de résistance à la collision, qui est demandée à une fonction de hashage dans le modèle computationnel : un adversaire polynomial ne peut pas trouver deux bitstrings différentes qui ont la même image par $\mathbf{hash}$, sauf avec une probabilité négligeable.

**Lemme 4.** *Soit $(\mathcal{C}_i)_{i\in I}$ une famille de contraintes générées par un adversaire, et telle que pour tout $i$, la complexité syntaxique de $\mathcal{C}_i$ est inférieure à $r$, et soit $(M_i, N_i)_{i\in I}$ une famille de termes telle que pour tout $i$, la complexité syntaxique de $M_i$ et de $N_i$ est inférieure à $r$, et tel que $f(|\ M\ |_l) = f(|\ N\ |_l)$*
*Soit $l$ un paramètre de sécurité. Soit $A \subseteq \left(\{0,1\}^{f(|M|_l)}\right)^2$. Alors il existe un polynome $P$ tel que :*

$$\sum_{1\leq i\leq n} \llbracket(\mathbf{hash}(M_i), \mathbf{hash}(N_i)) \in A \wedge not\,(M_i = N_i) \wedge \mathcal{C}_i\rrbracket_l \leq \#(A) \times \frac{P(r)}{2^{2\times f(|M|_l)}}$$

### 3.4.3 Egalité

On va maintenant définir : $[M = N \mid \mathcal{C}]_l$, qui correspond à la probabilité que les termes $M$ et $N$ soient égaux, sachant que la contrainte $\mathcal{C}$ est vérifiée :

**Definition 16.** *Soit $\mathbf{\Lambda}$ un langage, soit $\mathscr{M}$ une interprétation computationnelle des termes de $\mathbf{\Lambda}$, soit $l$ un paramètre de sécurité. On définit :*

$$[M = N \mid \mathcal{C}]_l = \frac{\mathscr{M}_l\left(\llbracket(M = N) \wedge \mathcal{C}\rrbracket_l\right)}{\mathscr{M}_l(\llbracket\mathcal{C}\rrbracket_l)}$$

**Exemple 17.** *On considère le langage **Strings**, et $\mathscr{M}$ la même interprétation computationnelle que celle utilisée à l'exemple 15. Soit $l \in \mathbb{N}$ un paramètre de sécurité. Alors :*
- $\left[\mathbf{concat}(\mathbf{un}, n^{(p\in\mathbb{N}\mapsto p)}) = \mathbf{concat}(m^{(p\in\mathbb{N}\mapsto 1)}, n^{(p\in\mathbb{N}\mapsto p)}) \mid \emptyset_{constraint}\right]_l = \frac{1}{2}$
- $\left[\mathbf{concat}(\mathbf{un}, \mathbf{zero}) = \mathbf{concat}(n^{(p\in\mathbb{N}\mapsto 1)}, m^{(p\in\mathbb{N}\mapsto 1)}) \mid \mathbf{zero} = m^{(p\in\mathbb{N}\mapsto 1)}\right]_l = \frac{\frac{1}{4}}{\frac{1}{2}} = \frac{1}{2}$

# 4 $\pi$-calcul appliqué avec contraintes

Depuis son introduction en [5], différentes approches mathématiques, ont été utilisées pour formaliser le modèle de sécurité symbolique. Le $\pi$-calcul appliqué, introduit par [1], est une extension du $\pi$-calcul [10] qui permet aux agents d'envoyer et de recevoir non seulement des names atomiques, mais également des termes formés à partir de fonctions et de names, et de tester l'égalité entre ces termes grâce à une théorie équationelle. Le $\pi$-calcul appliqué se prête naturellement à la modélisation de propriétés de sécurité sur les protocoles, par exemple les propriétés de secret ou d'indistinguabilité. On va définir dans cette section le $\pi$-calcul avec égalité probabiliste, qui étend la syntaxe et la sémantique du $\pi$-calcul appliqué.

## 4.1 Syntaxe des processus

On va définir *le $\pi$-calcul appliqué avec égalité probabiliste $\pi^{\mathcal{C}}$*, comme un langage de processus qui est construit à partir du $\pi$-calcul appliqué, en ajoutant des contraintes qui correpondent aux résultats de tous les tests qui ont été effectués avant le stade actuel de l'exécution.

On définit d'abord les *processus de base* , qui ne font pas intervenir de contraintes : il s'agit de la même grammaire que pour le $\pi$-calcul appliqué, avec la seule différence que l'on indice les names et les variables par une fonction de longueur :

**Definition 17.** *Soit $\mathbf{\Lambda} = (\Sigma, \mathcal{V}, \mathcal{N})$ un langage. L'ensemble des processus de base sur le langage $\mathbf{\Lambda}$ est donné par la grammaire suivante :*

$$P \ ::= \ 0 \ \Big| \ (P \mid Q) \ \Big| \ !\,(P) \ \Big| \ \nu n^{f} \cdot P \ \Big| \ \overline{p}\,(M) \cdot P \ \Big| \ p\left(x^{f}\right) \cdot P \ \Big| \ \textit{if } (M = N) \textit{ then } P \textit{ else } Q$$

*avec $n \in \mathcal{N}$, $f : \mathbb{N} \to \mathbb{N}$, $p \in \mathscr{N}$, $M, N \in$ Terms$(\mathbf{\Lambda})$, $x \in \mathscr{V}$*

La signification des processus est similaire à celle du $\pi$-calcul appliqué : Le processus nul $0$ ne fait rien, $(P \mid Q)$ est la composition parallèle de $P$ et $Q$, $!\,(P)$ est la réplication de $P$, qui se comporte comme un nombre infini de copies de $P$ en parallèle. $\nu n^{\mathrm{f}} \cdot P$ génère un nouveau name privé de longueur déclarée f, et ensuite exécute $P$, $\overline{p}\,(M) \cdot P$ est prêt à émettre le terme $M$ sur le canal $p$, puis à éxécuter le processus $P$, $p\left(x^{\mathrm{f}}\right) \cdot P$ est prêt à recevoir un terme sur le canal $p$, et ensuite à éxécuter $P$ où le terme reçu a remplacé la variable $x$. if $(M = N)$ then $P$ else $Q$ est le processus conditionnel qui se comporte comme $P$ si $M$ est égal à $N$ et comme $Q$ sinon : cependant l'égalité considérée ici est probabiliste, puisqu'il ne s'agit pas d'une égalité équationnelle, mais d'une probabilité, calculée sur les interprétations computationnelles, que deux termes soient égaux .

## 4.2 Processus étendus

On va maintenant étendre les processus, en y ajoutant : des substitutions actives, dont l'usage est analogue à celles du $\pi$-calcul appliqué, et des contraintes, qui correspondent à des restrictions des implémentations possibles à ce stade de l'exécution.

**Definition 18.** *Soit $\mathbf{\Lambda} = (\Sigma, \mathscr{N}, \mathscr{V})$ un langage. L'ensemble des processus étendu sur le langage $\mathbf{\Lambda}$ est donné par la grammaire suivante :*

$$A_1, A_2 \ ::= \ P \ \Big| \ \nu x^{\lambda} \cdot A_1 \ \Big| \ \nu n^{\lambda} \cdot \ \Big| \ \left\{\{M/x^{\lambda}\}\right\} \ \Big| \ (A_1 \mid A_2)$$

*où $P$ est un processus sur $\mathbf{\Lambda}$, $\lambda : \mathbb{N} \to \mathbb{N}$, et $M \in$ Terms$(\mathbf{\Lambda})$ tel que $\mid M \mid_l = \lambda(l)$*

**Exemple 18.** *On considère le langage **Strings**. Le processus étendu ci-dessous représente le processus qui teste si* $\mathbf{concat}(\mathbf{un}, \mathbf{zero})$ *est égal à* $\mathbf{zero}$*, qui s'arrete s'il y a égalité et qui émet* $\mathbf{concat}(\mathbf{un}, \mathbf{zero})$ *sinon.*

$$\left(\left\{\{\mathbf{concat}(\mathbf{un}, \mathbf{zero})/x^{(p \mapsto p)}\}\right\} \mid \textit{ if } \left(x^{(p \mapsto p)} = \mathbf{zero}\right) \textit{ then } 0 \textit{ else } \overline{p}\left(x^{(p \mapsto p)}\right) \cdot 0\right)$$

On va maintenant ajouter des contraintes aux processus. Les contraintes représentent le fait qu'il y a eu précédemment des tests d'égalité, et indiquent la manière dont ils ont été résolus pour cette trace particulière :

**Definition 19.** *Soit $\mathbf{\Lambda} = (\Sigma, \mathscr{N}, \mathscr{V})$ un langage L'ensemble $\pi^{\mathcal{C}}$ des processus étendus contraints est donné par la grammaire suivante :*

$$B_1, B_2 \ ::= \ A_1 \ \Big| \ \nu x^{\lambda} \cdot B_1 \ \Big| \ \nu n^{\lambda} \cdot B_1 \ \Big| \ (B_1 \mid B_2) \ \Big| \ \mathcal{C}$$

*où $A_1$ est un processus étendu, $x \in \mathscr{V}$, $n \in \mathscr{N}, \lambda : \mathbb{N} \to \mathbb{N}$, et $\mathcal{C} \in Constraints\,(\mathbf{\Lambda})$.*

**Exemple 19.** *On considère le langage **Strings**. Le processus ci-dessous est un processus étendu contraint.*

$$\Big( \textit{ if } \Big( n^{(p \mapsto p)} = m^{(p \mapsto p)} \Big) \textit{ then } P \textit{ else } Q \mid (\mathbf{concat}(\mathbf{un}, n^{(p \mapsto p)}) = \mathbf{concat}(\mathbf{un}, m^{(p \mapsto p)})) \Big)$$

*La contrainte représente le fait qu'un agent a déjà testé si $(\mathbf{concat}(\mathbf{un}, n^{(p \mapsto p)})$ était égal à $\mathbf{concat}(\mathbf{un}, m^{(p \mapsto p)}))$, et a trouvé que c'était vrai. L'évaluation du $if$ doit donc se faire de manière cohérente : pour l'interprétation "naturelle" du langage **Strings**, la condition doit être vraie avec probabilité 1.*

### 4.3 Sémantique du $\pi$-calcul appliqué avec contraintes

On veut donner une interprétation probabiliste à l'égalité, ce qui implique d'avoir une sémantique probabiliste. Une extension probabiliste du $\pi$-calcul appliqué, qui repose sur l'ajout d'un opérateur de choix probabiliste, a été introduit dans [7]. Les auteurs introduisent une sémantique probabiliste, et en déduisent un notion de probabilité sur les exécutions. Bien que notre but soit différent ici, les outils formels sont analogues.

**Congruence structurelle $\equiv$ :** On définit une relation de congruence structurelle sur le processus étendus contraints comme la plus petite relation qui vérifie les règles usuelles pour le $\pi$-calcul appliqué, auxquelles on ajoute la règle suivante (la totalité des règles est à la figure 6 de l'annexe D) : $(\mathcal{C}_1 \mid \mathcal{C}_2) \equiv \mathcal{C}_1 \wedge \mathcal{C}_2$

**Exemple 20.** *Comme on peut le voir sur l'exemple suivant, on peut regrouper en une seule contrainte des contraintes qui ont un champ d'application différent :*

$$(\nu n \cdot (\overline{p}\,(n) \cdot 0 \mid n = \mathbf{un})) \mid (n = \mathbf{zero}) \equiv \nu m \cdot ((\overline{p}\,(m) \cdot 0 \mid m = \mathbf{un} \wedge n = \mathbf{zero}))$$

**Relation de réduction interne probabiliste :** On définit les contextes d'évaluation $\mathscr{C}[\cdot]$, qui déterminent les contextes sous lesquels on peut appliquer les règles de réduction, de manière analogue au $\pi$-calcul appliqué.

Pour chaque paramètre de sécurité, on définit une relation de réduction. Puisque la sémantique est probabiliste, à un processus dans $\pi^{\mathcal{C}}$ on associe une distribution sur les classes de processus de $\pi^{\mathcal{C}}$ modulo la relation de congruence, c'est-à-dire une fonction : $\mu : \mathcal{P}\left(\pi^{\mathcal{C}} / \equiv\right) \to [0,1]$ qui est une mesure de probabilité sur $\mathcal{P}\left(\pi^{\mathcal{C}} / \equiv\right)$. On note cet ensemble $Distr(\pi^{\mathcal{C}})$. Etant donnés une distribution $\mu \in Distr(\pi^{\mathcal{C}})$, et $\mathscr{C}[\cdot]$ un contexte d'évaluation, on note $\mathscr{C}[\mu]$ la distribution définie par :

$$\mathscr{C}[\mu](P) = \begin{cases} \mu(Q) \text{ si } \exists Q \text{ tel que } P = \mathscr{C}[Q] \\ 0 \text{ sinon} \end{cases}$$

**Definition 20.** *Soit $\mathbf{\Lambda}$ un langage, $\mathscr{M} = (\mathscr{M}_l)_{l \in \mathbb{N}}$ une interprétation des termes de $\mathbf{\Lambda}$. Pour tout paramètre de sécurité $l \in \mathbb{N}$, on définit une relation de réduction probabiliste $\to^l$: ($Terms(\mathbf{\Lambda}) \to Distr(\pi^{\mathcal{C}})$), par les règles de la figure 2 :*

**Exemple 21.** *On considère le langage **Strings**, et son interprétation naturelle. Le protocole ci-dessous génère un bit, teste si il vaut $0$ ou $1$, puis ne fait rien si il vaut $1$, et émet sur le canal $p$ si il vaut $0$ :*

$$P \stackrel{def}{=} \nu n^{(p \mapsto 1)} \cdot \ \textit{ if } \Big( n^{(p \mapsto 1)} = \mathbf{un} \Big) \textit{ then } 0 \textit{ else } \overline{p}\,(\cdot) \cdot 0.$$

*Soit $l$ un paramètre de sécurité. Alors :* $P \to^l \ \frac{1}{2} \cdot \delta_{\nu n^{(p \mapsto 1)} \cdot 0 \mid (n^{(p \mapsto 1)} = \mathbf{un})} + \frac{1}{2} \cdot \delta_{\nu n^{(p \mapsto 1)} \cdot \overline{p}(\cdot) \cdot 0 \mid \textit{not}\left(n^{(p \mapsto 1)} = \mathbf{un}\right)}$

Egalité probabiliste : $(\text{ if } (M=N) \text{ then } P \text{ else } Q \mid \mathcal{C}) \to^{l} [M=N \mid \mathcal{C}]_l \cdot \delta_{(P \mid \mathcal{C} \wedge (M=N))} + (1-[M=N \mid \mathcal{C}]_l) \cdot \delta_{(Q \mid \mathcal{C} \wedge \text{not}(M=N))}$

Communication : $\left(\mathcal{C} \mid \left(\overline{p}\left(x^{\lambda}\right) \cdot P \mid p\left(x^{\lambda}\right) \cdot Q\right)\right) \to^{l} \delta_{(\mathcal{C} \mid (P \mid Q))}$ $\qquad \dfrac{B_1 \to^{l} \mu \qquad \mathscr{C}[\cdot] \text{ un contexte d'évaluation}}{\mathscr{C}[B_1] \to^{l} \mathscr{C}[\mu]}$ contexte

FIGURE 2 – Sémantique non labellée

$$\tau \, \frac{B_1 \to \mu}{B_1 \to^{\tau\, l} \mu}$$

$$(\mathcal{C} \mid \overline{p}\,(x) \cdot P) \to^{\overline{p}(x)\, l} \delta_{(\mathcal{C} \mid P)} \quad (\mathcal{C} \mid p\,(x) \cdot P) \to^{p(M)\, l} \delta_{(\mathcal{C} \mid P\{M/x\})}$$

$$\downarrow M = M_{\phi(A_1)} \text{ est un “ground terme”} \qquad \downarrow N = N_{\phi(A_1)} \text{ est un “ground terme”}$$

$$(\mathcal{C} \mid A_1) \to^{\text{test}(M,N)\, l} [\downarrow M =\downarrow N \mid \mathcal{C}]_l \cdot \delta_{(\mathcal{C} \wedge (\downarrow M = \downarrow N) \mid A_1)} + (1-[\downarrow M =\downarrow N \mid \mathcal{C}]_l) \cdot \delta_{(\mathcal{C} \wedge \text{not}(\downarrow M = \downarrow N) \mid A_1)}$$

FIGURE 3 – Sémantique labellée

**Sémantique labbellée :** On va maintenant définir la sémantique labellée $\to^{\alpha\, l}$, qui correspond à toutes les manières possibles dont le processus considéré, placé dans un contexte, pourrait évoluer. On la définit par les règles données en annexe D à la figure 7. La figure 3 présente certaines de ces règles. Les $\tau$-actions modélisent les actions internes au protocole, la règle **out-atom** modélise l'émission d'un message, la règle **in** modélise la réception d'un message, et la règle **test** modélise les tests que peut faire l'adversaire. Il faut ajouter à ces règles des règles de passage au contexte, qui sont les mêmes que pour le $\pi$-calcul appliqué.

## 4.4 Evolution probabiliste d'un processus

### 4.4.1 Exécution d'un processus

Pour la sémantique non labbellée, une exécution d'un processus est la donnée de tous les états par lesquels ce processus est passé, et de la distribution de probabilité associée à chaque transition. On note $B_1 \to_{\mu}^{l} B_2$, si $B_1 \to^{l} \mu$ et $\mu(B_2) > 0$.

**Definition 21.** *Pour un paramètre de sécurité $l$ fixé, une execution de $B_1$ est une suite finie ou infinie $B_1 = A_1 \to_{\mu_1}$ ,... $\to_{\mu_{n-1}}^{l} A_n$.... On note Exec$^{l}$ $(B_1)$ l'ensemble des exécutions du processus $B_1$.*

**Exemple 22.** *On considère le processus qui teste si deux names de longueur 1 sont égaux, et émet* $\mathbf{un}$ *sur le canal $p$ si c'est le cas, et émet* $\mathbf{zero}$ *(sur le même canal) sinon :*

$$P \stackrel{def}{=} \textit{ if } \left(n^{(p \mapsto 1)} = m^{(p \mapsto 1)}\right) \textit{ then } \overline{p}\,(\mathbf{un}) \cdot 0 \textit{ else } \overline{p}\,(\mathbf{zero}) \cdot 0$$

*On note $\mu = \frac{1}{2}\delta_{(\overline{p}(\mathbf{un}) \cdot 0)} + \frac{1}{2}\delta_{(\overline{p}(\mathbf{zero}) \cdot 0)}$. Pour un paramètre de sécurité quelconque $l \in \mathbb{N}$, les exécutions possibles de $P$ sont :*

$$P \qquad \textit{et} \qquad P \to_{\mu}^{l} \overline{p}\,(\mathbf{un}) \cdot 0 \textit{ et} \qquad P \to_{\mu}^{l} \overline{p}\,(\mathbf{zero}) \cdot 0$$

On définit de la même manière une exécution d'un processus pour la sémantique labbellée (G).
On cherche maintenant à exprimer la probabilité d'une exécution donnée. Pour cela, il faut d'abord résoudre entièrement le non-déterminisme, c'est-à-dire de spécifier un scheduler qui indique quelle règle appliquer à chaque stade de l'exécution.

### 4.4.2 Schedulers

Un scheduler est une fonction, qui, étant donnée une exécution, choisit une évolution possible du processus parmis toutes celles possibles, ou décide de ne pas évoluer : le but est de résoudre explicitement le non-déterminisme de la sémantique. Pour une execution $e$, on note last $(e)$ le dernier état de l'exécution.

**Definition 22.** *Un scheduler (pour la sémantique non labellée), est une fonction :*

$$\sigma : e \in \{\textit{executions}\} \to \mu \in \{\rho \mid \textit{last}\,((\,)\,e) \to \mu\} \cup \{\bot\}$$

**Exemple 23.** *On définit les processus suivants :*

$$Q_1 \stackrel{\textit{def}}{=} q\,(x) \cdot 0 \quad Q_2 \stackrel{\textit{def}}{=} \overline{q}\,(\cdot) \cdot 0 \qquad Q_3 \stackrel{\textit{def}}{=} \overline{q}\,(\cdot) \cdot \textit{ if } \left(n^{(p \mapsto 1)} = m^{(p \mapsto 1)}\right) \textit{ then } \overline{p}\,(\mathbf{zero}) \cdot 0 \textit{ else } 0 \quad P \stackrel{\textit{def}}{=} Q_1 \mid Q_2 \mid Q_3$$

*Un scheduler peut décider de synchroniser $Q_1$ et $Q_2$ ensemble, ou bien de synchroniser $Q_1$ avec $Q_3$, ou de ne rien faire. On va définir un scheduler $\sigma$ qui modélise la deuxième solution : soit $R =$ if $\left(n^{(p \mapsto 1)} = m^{(p \mapsto 1)}\right)$ then $\overline{p}\,(\mathbf{zero}) \cdot 0$ else $0 \mid Q_2$.*

$$\begin{aligned} \sigma(\{P\}) &= \delta_R \\ \sigma(\{P \to_{\delta_R} R\}) &= \frac{1}{2} \cdot \delta_{\overline{p}(\mathbf{zero}) \cdot 0 \mid Q_2} + \frac{1}{2} \cdot \delta_{Q_2} \\ \sigma(\cdot) &= \bot \textit{ sinon} \end{aligned}$$

On définit de manière analogue (G) un scheduler pour la sémantique labéllée (qu'on appellera stratégie dans la suite).

**Exemple 24.** *On considère le même processus $P$ qu'à l'exemple 23. On va maintenant donner une stratégie $\mathcal{S}$ pour la sémantique labéllée :*

$$\begin{aligned} \mathcal{S}(\{P\}) &= (\tau, \delta_R) \\ \mathcal{S}(\{P \to^{\tau}_{\delta_R} R\}) &= (\tau, \frac{1}{2} \cdot \delta_{\overline{p}(\mathbf{zero}) \cdot 0 \mid Q_2} + \frac{1}{2} \cdot \delta_{Q_2}) \\ \mathcal{S}(\{P \to^{\tau}_{\delta_R} R \to^{\tau}_{\left(\frac{1}{2} \cdot \delta_{\overline{p}(\mathbf{zero}) \cdot 0 \mid Q_2} + \frac{1}{2} \cdot \delta_{Q_2}\right)} Q_2\}) &= (\overline{q}(\cdot), \delta_0) \\ \mathcal{S}(\cdot) &= \bot \textit{ sinon} \end{aligned}$$

Etant donné un scheduler qui résout complètement le non déterminisme, on peut maintenant associer une probabilité aux exécutions. Dans ce but, pour chaque paramètre de sécurité $l$, pour chaque processus $B_1$, et chaque scheduler $\gamma$, on définit une $\sigma$-algèbre sur les exécutions du processus $B_1$ dirigées par $\gamma$ : $(\sigma\text{Field}^{\gamma}_{B_1})_l$, et une mesure de probabilité sur cete algèbre $\left(Prob^{\gamma}_{B_1}\right)_l$. Les détails de la construction se trouve en annexe G.

**Exemple 25.** *On s'intéresse aux exécutions du processus $P$ dirigées par le scheduler $\sigma$ défini à l'exemple 23. Alors pour tout $l \in \mathbb{N}$,*

$$(Prob^{\sigma}_{P})_l \left(\{P \to_{\delta_R} R \to_{\frac{1}{2} \cdot \delta_{\overline{p}(\mathbf{zero}) \cdot 0 \mid Q_2} + \frac{1}{2} \cdot \delta_{Q_2}} Q_2\}\right) = \frac{1}{2}$$

On peut maintenant exprimer la probabilité que, pour un paramètre de sécurité $l \in \mathbb{N}$, le processus $P$ dirigé par le scheduler $\sigma$, évolue en un nombre borné d'étapes en un processus $Q$ qui appartient à un ensemble $E$ fixé.

$$\mathcal{P}_l \left(B_1 \stackrel{\sigma}{\to}^{\leq n} E\right) = (Prob^{\sigma}_{P})_l \left(\left\{e \in \text{Exec}\,(B_1, \sigma) \mid e^i \in H \text{ for some } i \leq n\right\}\right)$$

.

**Exemple 26.** *On considère le processus $P$ et le scheduler $\sigma$ de l'exemple 25. On voit que, lorsque l'évolution du processus est dirigée par le scheduler $\sigma$, on a une probabilité $\frac{1}{2}$ d'obtenir le processus $P_2$ en deux étapes de réduction. Alors si on prend comme ensemble d'états d'arrivée l'état $Q_2$, comme borne $n \geq 2$, on obtient :*

$$\mathcal{P}_l \left(B_1 \stackrel{\sigma}{\to}^{\leq n} \{Q_2\}\right) = \frac{1}{2}$$

De la même manière, pour la sémantique labellée, on définit $\mathcal{P}_l \left(B_1 \stackrel{\mathcal{S}}{\to}^{\leq n} E\right)$ comme la probabilité que, pour un paramètre de sécurité $l$, étant donnée une stratégie $\mathcal{S}$, le processus $P$ dirigé par la stratégie $\mathcal{S}$ évolue par la sémantique labellée en un nombre d'étapes inférieur à $n$ en un état qui appartient à $E$ :

# 5 Langage pour les distances-bouding protocoles

On va maintenant utiliser les outils introduits précedemment pour étudier des protocoles de distance bounding, c'est-à-dire des protocoles dont le but est de certifier que deux agents sont proches l'un de l'autre. La particularité de ces protocoles est d'avoir des phases rapides, durant lesquelles l'un des agents envoie un message, et mesure le temps qui s'écoule avant de recevoir la réponse. Si ce temps est supérieur à une valeur donnée, le protocole échoue.

On modélise cette situation de la manière suivante : On introduit un ensemble des localisations $Loc$ a vocation à représenter la localisation des processus : on peut voir $w_1 \in Loc$ comme un marqueur qui peut être possédé seulement par les processus qui se trouve dans la sphère de centre $w_1$ et de rayon une constante fixée. Sur la figure de gauche ci-dessous, le protocole modélisant l'agent $\text{Agent}_1$ ne peut pas utiliser la location $w_2$, et le protocole modélisant l'agent $\text{Agent}_2$ ne peut pas utiliser la location $w_1$. Sur la figure de droite ci-dessous, le protocole modélisant l'agent $\text{Agent}_1$ ne peut pas utiliser la location $w_2$, et le protocole modélisant l'agent $\text{Agent}_2$ ne peut pas utiliser la location $w_1$, tandis que le protocole modélisant l'agent $\mathscr{A}$, peut utiliser les locations $w_1$ et $w_2$.


On considère qu'il y a deux modes de communication : un mode normal et un mode rapide. Le mode rapide correspond aux communications qui se déroulent entre le moment où un agent a enclenché son chronomètre, et le moment où le temps maximum acceptable pour la communication est dépassé. Puisqu'on s'intéresse au temps de propagation du signal, on devrait considérer des communications asynchrones. Néanmoins, on modélise cette situation par des communications synchrones, ce qui implique en contre partie une disymétrie entre la réception et l'émission de messages : en effet, lorsque la phase rapide est déclenchée par l'agent A :

- dans la réalité, les agents *loin* de l'agent A ne peuvent pas recevoir les messages envoyés par les agents *près* de l'agent A avant la fin de la phase rapide (à cause du temps de propagation du signal). Néanmoins, après la fin de la phase rapide, les agent lointains ont accès au messages qui ont été émis pendant la phase rapide. On modélise cela par le fait que les agents lointains peuvent recevoir des messages dans la phase rapides, mais ils ne peuvent pas les utilisés avant la fin de la phase rapide.
- dans la réalité, les agent *loin* de l'agent A ne peuvent pas, après le début de la phase rapide, envoyer de messages qui seront reçus avant la fin de la phase rapide. Néanmoins, ils pourraient les envoyer avant l'enclenchement de la phase rapide, et que ces messages soient reçus avent la fin. On modélise cela par le fait que **tous** les agents peuvent émettre des messages pendant la phase rapide (puisqu'ils n'ont reçu aucune information depuis l'enclenchement de la phase rapide, c'est équivalent à considérer qu'ils envoient les messages en avance).
- les phases rapides déclenchées par des agents à des localisations différentes sont disjointes.

## 5.1 Syntaxe

**Syntaxe des processus :** On ajoute des primitives supplémentaires au langage de processus de la définition 17 pour modéliser les phases rapides.

**Definition 23.** *Soit $\mathbf{\Lambda} = (\Sigma, \mathcal{V}, \mathcal{N})$ un langage, et $Loc = \{w_1, w_2, ...\}$ un ensemble dénombrable de localisations. L'ensemble des processus sur le langage $\mathbf{\Lambda}$ et les localisations Loc est donné par la grammaire donnée précedemment à la définition 17, à laquelle on ajoute les règles suivantes :*

$$P ::= \cdots \mid \overline{\mathit{fast}(p, w_1)}\,(M) \cdot P \mid \mathit{fast}(p, w_1)\,(x^f) \cdot P \mid \mathit{begin}\,(w_1) \cdot P \mid \mathit{end} \cdot P \mid \mathit{wait}^{w_1}\,(X) \cdot P$$

*où $p \in \mathcal{N}$, $w_1 \in Loc$, $M \in \mathit{Terms}(\mathbf{\Lambda})$, $x \in \mathcal{V}$, $f : \mathbb{N} \to \mathbb{N}$ et $X \subseteq \mathcal{V}$.*

Les primitives begin $() \cdot ()$ et end$\cdot$ servent à déclencher et à terminer une phase rapide : cela correspond au fait qu'un agent déclenche son chronomètre, et ne prend en compte que les intéractions qu'il a avant qu'un temps fixé ne soit écoulé. Les cannaux fast$(p, w_1)$ sont les seul cannaux qui peuvent être utilisés dans les phases rapides. La primitive $wait$ sert à modéliser le fait qu'un message reçu par un agent ne peut pas être utilisé immédiatement : wait$^{w_1}(x) \cdot P$ signifie que toute action du processus $P$ qui utilise $x$ sera bloquée jusqu'à la fin de la phase rapide.

**Exemple 27.** *On considère le langage **Strings**, et $Loc = w_1, w_2$. on définit :*

$$P \stackrel{def}{=} \mathit{begin}\,(w_1) \cdot \overline{\mathit{fast}(w_1, p)}\,(\mathbf{un}) \cdot \mathit{fast}(w_1, p)\,(x) \cdot \mathit{end} \cdot 0.$$

*$P$ déclenche une phase d'échange rapide, émet le terme $un$ sur le canal $p$, recoit un terme sur ce même canal, et termine la phase d'échange rapide.*

On impose des restrictions sur la syntaxe des processus pour assurer qu'une phase rapide ne peut être terminée que par le processus qui l'a déclenchée : on dit qu'un processus est bien formé, si
- Dans chaque branche de l'arbre de syntaxe, chaque begin $(\cdot) \cdot$ doit être suivi par exactement un end$\cdot$
- Dans un processus begin $(w_1) \cdot P$, $P$ ne doit contenir ni réplication ni composition parallèle.

Dans la suite, tous les processus considérés sont bien formés.

**Marqueurs des phases rapides** A chaque point de l'execution, on veut savoir si une phase rapide est en cours, et si oui à quelle localisation est l'agent qui l'a déclenchée. Pour cela, on introduit des *locks sur les localisations* :

$$\mathbf{l}_1 \ ::= \ \top \mid \text{lock}\,(w_1^n) \ \text{avec } n \geq 1,\ w_1 \in Loc$$

$\top$ indique qu'aucune phase rapide n'est en cours, lock $(w_1^n)$ indique qu'une phase rapide est en cours, et que $n$ agents à la localisation $w_1$ y participent.

## 5.2 Sémantique

**Sémantique non labellée :** Pour la sémantique non labellée, un état est un couple formé d'un processus et d'un lock sur les localisations qui indique les phases rapides en cours. La règle suivante exprime que la sémantique non labellée étend la relation $\rightarrow^l$ du $\pi$-calcul avec égalité probabiliste par la règle suivante :

$\dfrac{B_1 \rightarrow^l \mu}{B_2, \mathbf{l}_1 \rightarrow^l \mu, \mathbf{l}_1}$ avec $(\mu, \mathbf{l}_1)$ la distribution définie par : $(\mu, \mathbf{l}_1)(P, \mathbf{l}_2) = \mu(P)$ si $\mathbf{l}_1 = \mathbf{l}_2$, 0 sinon.

On y ajoute des règles spécifique à l'échange rapide (les détails se trouvent à la figure 8 de l'annexe E).

**Exemple 28.** *On s'intéresse au processus $P$ défini à l'exemple 27. L'agent situé à la localisation $w_1$ déclenche une phase rapide : le lock sur les localisations passe de $\top$ à lock $\left(w_1{}^1\right)$ :*

$$\mathit{begin}\,(w_1) \cdot \overline{\mathit{fast}(w_1, p)}\,(\mathbf{un}) \cdot \mathit{fast}(w_1, p)\,(x) \cdot \mathit{end} \cdot 0, \top \rightarrow \delta_{\left(\overline{\mathit{fast}(w_1,p)}(\mathbf{un}) \cdot \mathit{fast}(w_1,p)(x) \cdot \mathit{end} \cdot 0, \mathit{lock}(w_1{}^1)\right)}$$

**Sémantique labellée :** Pour la sémantique labellée, un état est un triplet formé d'un processus, d'un lock sur les localisations, et d'un ensemble de variables qui ne peuvent etre connues que dans la sphère correspondant à la localisation.

On définit la notion de scheduler pour la sémantique non labellée, et de stratégie pour la sémantique labellée, de manière analogue au $\pi$-calcul avec égalité probabiliste (la définition précise se trouve à la figure 10 de l'annexe E).

## 5.3 Modélisation de l'adversaire

En fonction du type d'attaque que l'on veut considérer, un adversaire est autorisé à utiliser un ensemble fixé de localisations, et à intéragir avec un ensemble fixé de processus, qui correspond à tous les agents avec lesquels l'adversaire peut intéragir. Un adversaire est représenté par un processus $\mathcal{A}_1$ (sans contraintes, ni déclenchement ou achèvement d'un

échange rapide). On s'intéresse à des propriétés d'atteignabilité : une attaque contre un ensemble d'agents modélisé par le processus $P$, s'exprime comme la probabilité d'atteindre un certain ensemble $E$ d'états (qui dépend de l'attaque qu'on veut considérer), en partant de l'état $(P \mid \mathcal{A}_1, \top)$. On veut considérer des ensembles $E$ caractérisés par l'ensemble des états où le protocole s'est terminé avec succès. Pour cela, on dispose d'un cannal **result** dont l'adversaire ne dispose pas, et qui ne lui est jamais envoyé. On définit donc $Process^{\mathbf{result}}$ comme l'ensemble des protocoles qui n'émettent jamais de termes dans lequel **result** apparaît, et $Adv_L^{\mathbf{result}}$ l'ensemble des processus modélisant un adversaire $\mathcal{A}_1$ tel que **result** $\notin fn(\mathcal{A}_1)$, et l'adversaire n'utilise que des localisations de $L$. On définit l'ensemble $E$ des états caractérisant l'attaque comme l'ensemble des états $(Q, \mathbf{l}_1)$, tel que $Q$ est prêt à émettre $\mathbf{un}$ sur le canal **result**. On considère que l'adversaire peut choisir le scheduler qui l'avantage le plus. Pour un adversaire $\mathcal{A}_1$ fixé, la probabilité de l'attaque en un nombre d'étape inférieur à $n$ est donnée par :

$$\max_{\sigma \text{ a scheduler}} \mathcal{P}_l\left(B_1 \mid \mathcal{A}_1, \top \xrightarrow{\sigma}{}^{\leq n} E\right)$$

. On veut pouvoir caractériser l'adversaire sans avoir à considérer tous les processus possibles modélisant un adversaire. Pour cela, on va caractériser l'adversaire et le scheduler qu'il contrôle par la donnée d'une stratégie pour la sémantique labellée. De manière analogue avec le cas de la sémantique non labellée, on définit $Strat_L^{\mathbf{result}}$ comme l'ensemble des stratégies qui ne disposent que des localisations $L$, et n'utilisent pas le canal **result**.

Le théorème suivant exprime le fait qu'on peut exprimer de manière équivalente l'adversaire comme un couple formé d'un processus et d'un scheduler pour la sémantique non labellée, ou comme une stratégie pour la sémantique labellée.

**Théorème 1.** *Soit* ***result*** *un canal. On définit* $E = \{(B_2, \mathbf{l}_1) \mid B_2 \equiv (\overline{\boldsymbol{result}}(\mathbf{un}) \cdot B_3 \mid B_4)\}$. *On définit* $E_{lab} = \{(B_2, V, \mathbf{l}_1) \mid B_2 \equiv \overline{\boldsymbol{result}}(\mathbf{un}) \cdot B_3 \mid B_4\}$.
*Alors pour tout paramètre de sécurité* $l$, *pour toute borne de sécurité* $n$, *pour tout processus étendu contraint* $B_1 \in Process^{\boldsymbol{result}}$, *pour tout ensemble de localisations* $L$ *:*

$$\max_{\mathcal{S} \in Strat_L^{\boldsymbol{result}}} \mathcal{P}_l\left(B_1, \emptyset, \top \xrightarrow{\mathcal{S}}{}^{\leq n} E_{lab}\right) = \max_{\mathcal{A}_1 \in Adv_L^{\boldsymbol{result}}} \max_{\sigma \text{ un scheduler}} \mathcal{P}_l\left(B_1 \mid \mathcal{A}_1, \top \xrightarrow{\sigma}{}^{\leq n} E\right)$$

La preuve de ce théorème est détaillée dans l'annexe H.

# 6 Etude de cas

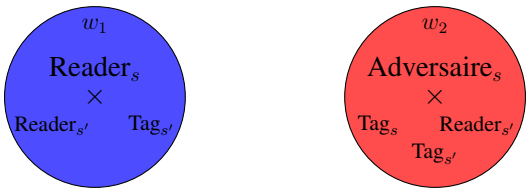
Un protocole distance bounding est la donnée d'un protocole pour un agent appelé Reader, d'un protocole pour un agent appelé Tag, qui paratgent (au moins) une clé secrète, visant à ce que Reader certifie que Tag est proche de lui.

## 6.1 Type d'attaque spécifique aux distance-bounding protocole

**Attaque à distance :** On considère la situation suivante : un adversaire cherche à faire croire à un lecteur éloigné $R$, avec qui il partage la même clé secrète $s$, qu'il est proche de lui.

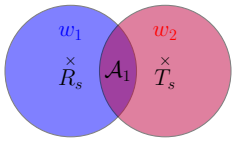
L'attaque comporte en fait deux phases : Dans une première phase d'apprentissage, l'adversaire, le lecteur cible, et un tag correspondant à ce lecteur sont placés au même endroit. Dans une deuxième phase, l'adversaire est loin du lecteur cible, et aucun tag correspondant au lecteur cible n'est proche de ce lecteur. L'adversaire cherche alors à obtenir que le protocole correspondant au lecteur termine avec succès. Pendant les deux phases, des lecteurs et tags honnêtes (avec une clé $s' \neq s$) peuvent se trouver proches du lecteur Reader, proches de l'adversaire, et plus globalement n'importe où dans l'espace. Des tag honnêtes, avec la même clé que le lecteur cible, peuvent également se trouver n'importe où dans l'espace sauf près du lecteur.

La deuxième phase peut être représentée par :



**Attaque mafia :** On considère la situation suivante : un lecteur $R$ et le tag correspondant $T$ (qui partagent une clé secrète $s$) sont éloignés l'un de l'autre. Un adversaire $\mathcal{A}_1$, qui ne possède pas la clé secrète $s$, et qui est près à la fois de $T$ et de $R$, cherche à faire croire à $R$ que $T$ est près de lui.

Comme précédemment, l'attaque comporte en fait deux phases : une première phase d'apprentissage, où l'adversaire, le lecteur cible, et un tag correspondant à ce lecteur sont placés au même endroit, et une deuxième phase, où $R$ et $T$ sont éloignés l'un de l'autre. L'adversaire cherche alors à obtenir que le protocole correspondant au lecteur termine avec succès. Comme précédemment, pendant les deux phases, des lecteurs et tags honnêtes (avec une clé $s' \neq s$) peuvent se trouver proches de $R$, proches de $T$ ou proches de l'adversaire, et plus globalement n'importe où dans l'espace. Des tags honnêtes, avec la même clé que le lecteur cible, peuvent également se trouver n'importe où dans l'espace sauf près du lecteur. La deuxième phase peut être représentée par le shéma ci-dessous (par souci de simplicité, les agents témoins n'apparaissent pas).




## 6.2 Protocole proposé par Tu et Piramithu

Le protocole de Tu et Piramithu (présenté en introduction) présente une attaque mafia, de probabilité 1, qui s'éxécute en temps linéaire. En effet, l'attaque contre le secret de la clé $s$ présentée en introduction peut aussi être vue comme une attaque mafia : l'adversaire apprend tous les bits de la clé $s$ les uns après les autres, puis simule le protocole d'un tag légitime. On va montrer que cette attaque est exprimable dans notre langage : En effet, on peut exprimer ce protocole par un processus $R_{s,w_1}$ correspondant au lecteur (où $s$ représente la clé secrète, et $w_1$ la localisation du lecteur) et un processus $T_{s,w_1}$ correspondant au tag (où $s$ représente la clé secrète du tag, et $w_1$ sa localisation) (les détails sont donnés en annexe). On s'intéresse à l'état de départ

$$\text{start} = \nu s \cdot (T_s^{w_1} \mid R_s^{w_2} \mid !\,((T_s^{w_3} \mid R_s^{w_3}\{n/\textbf{result}\})))$$

(avec $w_1 \neq w_2$), qui exprime le fait que le tag et le lecteur partagent la même clé, mais sont placés à des localisations différentes. Le fait d'ajouter $!\,((T_s^{w_3} \mid R_s^{w_3}\{p/\textbf{result}\}))$ à l'état de départ permet d'exprimer que l'adversaire peut intéragir avec un nombre non borné de tags et de lecteurs témoins (de clé secrète $s$), qui sont loin à la fois du tag cible et du lecteur cible. Le fait qu'on remplace le canal **result** par le cannal $p$ permet de modéliser le fait que ce sont des tags témoins. L'attaquant à accès à toutes les localisations. Soit $l$ un paramètre de sécurité. On peut alors construire un adversaire $\mathcal{A}_1$ tel que :

$$\max_{\sigma \text{ un scheduler}} \mathcal{P}_l\left(\text{start} \mid \mathcal{A}_1 \xrightarrow{\sigma}{}^{\leq 10\cdot l^2} E\right) = 1,$$

où $E$ est l'ensemble des états prêts à émettre **un** sur le cannal **result**.

## 6.3 Hancke- Kuhn Protocole

**Protocole :** On considère le protocole suivant (voire figure 4) : le lecteur $R$ et le tag $T$ s'échangent des nonces, puis calculent chacun $a = h(s, N_R, N_T)$. Ils coupent ensuite $a$ en deux bitstrings de longueur égales : $a = a_1 :: a_2$. Ensuite, le lecteur déclenche une phase rapide, et envoie un nonce $k$ au tag. Le protocole s'achève avec succès si le tag répond par $F(k, a_1, a_2)$ dans le temps imparti, et échoue sinon.

On peut modéliser ce protocole dans notre langage, par un protocole $R_{s,w_1}$ qui correspond au lecteur, et un protocole $T_{s,w_1}$ qui correspond au tag (voir détails en annexe).

**Sécurité du protocole :** La modélisation de l'attaque à distance correspond à deux phases : une phase d'apprentissage où l'adversaire a accès à des tags et des lecteurs à la même localisation que lui, et une deuxième phase pendant laquelle il lance son attaque. On va ici donner encore plus de pouvoirs à l'adversaire : on suppose qu'il connaît tous les secrets de

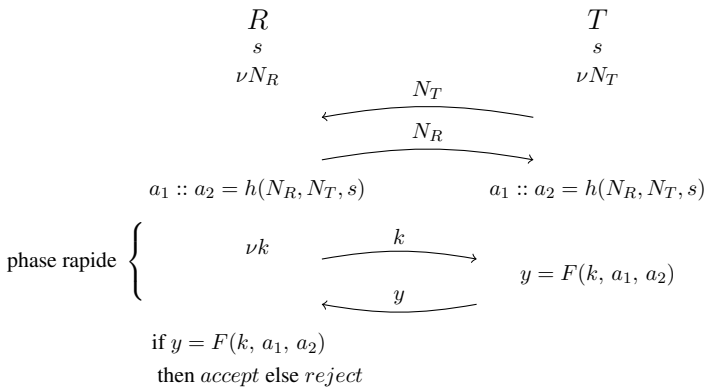



FIGURE 4 – Protocole Distance-Bounding proposé par Hancke et Kuhn

tous les protocoles : la phase 1 devient alors inutile, puisqu'il peut la simuler par lui-même. De meme, puisque l'adversaire peut simuler tous les agents qui ne sont pas à la localisation $w_1$, il est inutile de les ajouter au système.
On peut donc considérer l'état de départ suivant :

$$\text{start} = R_{s',w_1} \mid ! \left(R_{s_2,w_1}\{p/\text{result'}\}\right) \mid ! \left(T_{s_2,w_1}\right)$$

L'attaquant a donc accès à un lecteur cible à la localisation $w_1$, et des lecteurs et tags témoins à la localisation $w_1$. L'attaquant lui-même n'a pas accès à la localisation $w_1$.

Pour tout paramètre de sécurité $l$, pour toute attaque en temps polynomial en $l$, et tel que l'adversaire ne produit que des termes de complexité syntaxique de longueur polynomiale en $l$, il existe un polynome $P$ tel que la probabilité de l'attaque est inférieure à $P(l) \cdot (\frac{3}{4})^l$.

On utilise l'équivalence entre la sémantique non labellée et la sémantique labellée. On note $E_{\text{lab}}$ l'ensemble des processus prêts à émettre **un** sur le canal **result**. On montre (la preuve est en annexe) :

**Théorème 2.** *Pour tout paramètre de sécurité $l$, pour tout polynome $Q$, et pour toute stratégie $\mathcal{S} \in Strat_{Loc\setminus\{w_1\}}$, qui n'utilise pas le canal **result** et qui n'utilise que des termes de complexité syntaxique au plus $Q(l)$, il existe un polynome $P$, tel que :*

$$\mathcal{P}_l\left(\textit{start} \overset{\mathcal{S}}{\rightarrow}^{\leq Q(l)} E_{\textit{lab}}\right) \leq P(l) \cdot \left(\frac{3}{4}\right)^l$$

# 7 Conclusion

Dans le cadre de ce travail, nous avons défini un modèle de sécurité hybride entre le modèle symbolique et le modèle computationnel, nous en avons donné une formalisation opérationnelle sous la forme d'une extension probabiliste du $\pi$-calcul. Nous avons ensuite adapté ce langage au cas particulier des protocoles distance-bounding, et nous l'avons utilisé pour exprimer une attaque sur le protocole de Tu et Piramuthu, et donner une borne supérieure sur la probablité de succès d'une attaque à distance sur le protocole d'Hancke Kuhn.
Il serait intéressant d'étudier dans ce modèle des protocoles plus compliqués, et d'envisager une automatisation partielle des preuves. On pourrait également explorer les liens entre ce modèle et les modèles computationnels et symboliques : en particulier, on pourrait étudier la question de l'existence d'un ensemble minimal de constructeurs tels qu'on puisse simuler le fonctionnement d'une machine de Turing probabiliste en temps polynomial, par un processus de ce langage évoluant en temps plynomial.

# A

# Références


[1] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. *SIGPLAN Not.*, 36(3) :104–115, January 2001.

[2] David Basin, Srdjan Čapkun, Patrick Schaller, and Benedict Schmidt. Let's get physical : models and methods for real-world security protocols. In *Proceedings of the 22nd TPHols (International Conference on Theorem Proving in Higher Order Logics)*, 2009.

[3] Bruno Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 2001. IEEE Computer Society.

[4] Stefan Brands and David Chaum. Distance-bounding protocols (extended abstract). In *EUROCRYPT93, Lecture Notes in Computer Science 765*, pages 344–359. Springer-Verlag, 1993.

[5] Danny Dolev and Andrew Chi chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29 :198–208, 1983.

[6] Saar Drimer and Steven J. Murdoch. Keep your enemies close : distance bounding against smartcard relay attacks. In *USENIX Security Symposium*, pages 87–102, August 2007.

[7] Jean Goubault-Larrecq, Catuscia Palamidessi, and Angelo Troina. A probabilistic applied pi–calculus. In Zhong Shao, editor, *Programming Languages and Systems*, volume 4807 of *Lecture Notes in Computer Science*, pages 175–190. Springer Berlin Heidelberg, 2007.

[8] Gerhard P. Hancke and Markus G. Kuhn. An RFID Distance Bounding Protocol. In *International Workshop on Security*, 2005.

[9] Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. Information security and cryptology — icisc 2008. chapter The Swiss-Knife RFID Distance Bounding Protocol, pages 98–115. Springer-Verlag, Berlin, Heidelberg, 2009.

[10] Robin Milner. *Communicating and Mobile Systems : The &Pgr;-calculus*. Cambridge University Press, New York, NY, USA, 1999.

[11] Dusko Pavlovic and Catherine Meadows. Bayesian authentication : Quantifying security of the hancke-kuhn protocol. *Electr. Notes Theor. Comput. Sci.*, 265 :97–122, 2010.

[12] Yu-Ju Tu and Selwyn Piramuthu. RFID Distance Bounding Protocols. In *First International EURASIP Workshop on RFID Technology*, Vienna, Austria, September 2007.

# B Attaque sur le protocole de Tu et Piramithu

L'attaque sur le protocole de Tu et Piramithu peut etre représenté par la figure suivante :

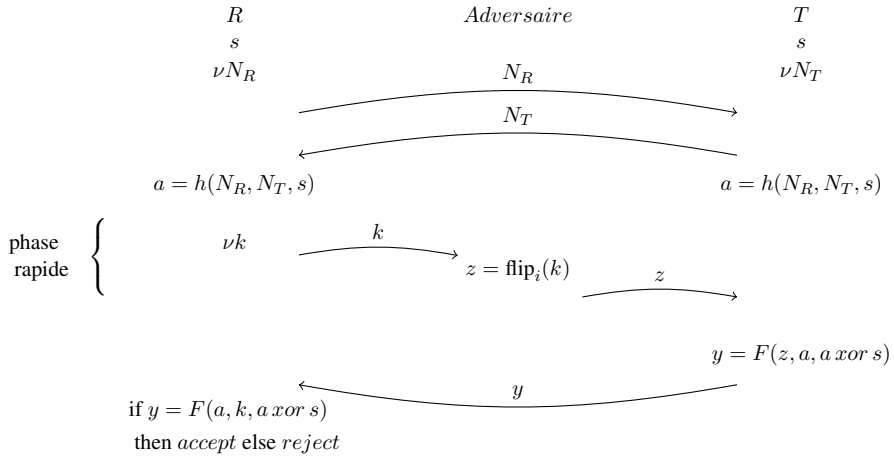



FIGURE 5 – Attaque sur le protocole de Tu et Pitramithu

# C Rappel mathématiques sur les $\sigma$-algèbres

Pour $X$ un ensemble, on note $\mathcal{P}(X)$ l'ensemble des parties de $X$.

**Definition 24.** *Soit $X$ une ensemble. $Alg \subseteq \mathcal{P}(X)$ est une $\sigma$ algèbre sur $X$ si :*
- *$Alg$ est non vide*
- *$Alg$ est clos par complémentation*
- *$Alg$ est clos par unions dénombrables*

**Definition 25.** *Soit $X$ un ensemble. Soit $Alg$ une $\sigma$-algèbre sur $X$. Une mesure de probabilité $\mathscr{D}$ sur $Alg$ est une fonction $Alg \to [0,1]$ telle que :*
- *$\forall Y \in Alg, \mathscr{D}(Y) \geq 0$*
- *$\mathscr{D}(\emptyset) = 0$*
- *$\mathscr{D}(\bigcup_{i\in\mathbb{N}} A_i) = \sum_{i\in\mathbb{N}} \mathscr{D}(A_i)$ si les $A_i$ sont des éléments de $Alg$ deux à deux disjoints.*
- *$\mathscr{D}(X) = 1$*

# D $\pi$-calcul appliqué avec égalité probabiliste et sécurité

## D.1 Définition de l'indice maximale et de la complexité syntaxique d'un terme

Les définitions exprimés intuitivement dans le corps du mémoire peuvent être formalisées de la manière suivante :

**Definition 26.** *Soit $\mathbf{\Lambda}$ un langage, $M \in Terms(\mathbf{\Lambda})$, $l \in \mathbb{N}$ un paramètre de sécurité. On définit l'indice maximal de $M$ pour $l$ par :*

$$LMax(M)_l = max\{|\ N\ |_l \mid N \text{ sous-terme de } M\}$$

**Definition 27.** *On associe à chaque terme $M \in Terms(\mathbf{\Lambda})$ une complexité syntaxique, qui est égale au nombre total d'occurences de constructeurs dans ce terme.*

## D.2 Extension d'une mesure sur le semi-anneau des implémentations partielles de la signature à l'algèbre des implémentations de la signature

On doit montrer que, pour définir une mesure de probabilité sur la $\sigma$-algèbre des implémentations de la signature, il suffit de donner la valeur de cette mesure sur les ensemble d'implémentations dirigées par une certaine implémentation partielle $\eta$.

On définit ici rigoureusement l'ensemble des implémentations dirigées par une implémentation partielle :

**Definition 28.** *Soit $\eta = (\eta_1, ..., \eta_n) \in \mathit{ImplSig}_m(\mathbf{\Lambda})$. On définit l'ensemble des implémentations dirigées par $\eta$ : $\uparrow^{sig}(\eta)_m^{\mathbf{\Lambda}}$ comme étant l'ensemble des éléments $f = (f_1, ..., f_n) \in \mathit{ImplSig}(\mathbf{\Lambda})$ tel que pour tout $j \in \{1, .., n\}$, la restriction de $f_j$ à $\{0,1\}^{\leq m}$ est égale à $\eta_j$ : $\uparrow^{sig}(\eta)_m^{\mathbf{\Lambda}} = \{f = (f_1, ..., f_n) \in \mathit{ImplSig}(\mathbf{\Lambda}) \mid \forall j \in \{1, .., n\}, \forall x \in \{0,1\}^{\leq m},\ f_j(x) = \eta_j(x)\}$*

Il s'agit en fait de démontrer le lemme suivant :

**Lemme 5.** *On définit :*
*$S \stackrel{def}{=} \emptyset \bigcup \{\uparrow^{sig}(\eta)_m^{\mathbf{\Lambda}}$ tel que $m \in \mathbb{N}$, $\eta \in \mathit{ImplSig}_m(\mathbf{\Lambda})\}$*
*Soit $\mathscr{D} : S \to [0,1]$ qui vérifie :*
- *$\forall Y \in S, \mathscr{D}(Y) \geq 0$, $\mathscr{D}(\emptyset) = 0$*
- *Si $A \in S$, et $A = \bigcup_{i \in \mathbb{N}} A_i$), et les $A_i$ sont des éléments de S deux à deux disjoints, alors : $\mathscr{D}(A) = \sum_{i \in \mathbb{N}} \mathscr{D}(A_i)$.*
- *$\mathscr{D}(\mathit{ImplSig}(\mathbf{\Lambda})) = 1$ (On remarque que $\mathit{ImplSig}\mathbf{\Lambda} \in S$, car cela correspond aux implémentations dirigées pour une implémentation partielle pour $m = 0$).*

*Alors il existe une unnique mesure de probabilité sur $AlgSig_{\mathbf{\Lambda}}$ qui coincide avec $\mathscr{D}$ sur son domaine de définition.*

*Démonstration.* On va utiliser le théorème d'extension de Carathéodory. Pour cela, il suffit de vérifier que l'ensemble S est un semi-anneau d'ensemble, c'est-à-dire :
- $\emptyset \in$ S
- Pour $A, B \in$ S, on a $A \cap B \in$ S
- Pour $A, B \in$ S, on a : $A \setminus B$ qui peut s'écrire comme union finie disjointe d'éléments de S : il existe $m \in \mathbb{N}$, et une famille $(K_i)_{i \leq m}$ deux à deux disjoints tels que $A \setminus B = \bigsqcup_{1 \leq i \leq m} K_i$

Ces conditions étant réunies, le théorème d'extension de Carthéodory indique que : toute pré-mesure $\mu$ sur le semi-anneau S admet une mesure sur la $\sigma$-algèbre généré par S dont la restriction à S est égale à $\mu$. De plus, si $\mu$ est $\sigma$-finie (en particulier dans le cas qui nous intéresse si $\mu(\text{ImplSig}\mathbf{\Lambda}) = \mu(\uparrow^{\text{sig}}(\cdot)_0^{\mathbf{\Lambda}}) = 1$, l'extension est unnique.

□

## D.3 Construction de la $\sigma$-algèbre des implémentations des names

Etant donné un paramètre de sécurité $l \in \mathbb{N}$, une interprétation des names pour $l$ est une fonction : $\Theta : \mathscr{N} \times (\mathbb{N} \to \mathbb{N}) \to (\{0,1\}^\star)$, pour $l \in \mathbb{N}$, qui vérifient la condition suivante : pour tout paramètre de sécurité $l$, pour tout $n \in \mathscr{N}$, et f : $\mathbb{N} \to \mathbb{N}$, la longueur de $\Theta(n, \text{f})$ est égale à la valeur de f appliquée au paramètre de sécurité $l$.

**Definition 29.** *Soit $\mathbf{\Lambda} = \Sigma, \mathscr{V}, \mathscr{N}$ un langage.*
*On définit :*

$$\mathit{ImplNames}(\mathbf{\Lambda}, l) = \{\Theta : \mathscr{N} \times (\mathbb{N} \to \mathbb{N}) \to \{0,1\}^\star \text{ tel que } \forall n \in \mathscr{N}, f : \mathbb{N} \to \mathbb{N},\ |\Theta(n,f)| = f(l)\}$$

.

**Exemple 29.** *On considère $\mathbf{\Lambda} = \Sigma, \mathscr{V}, \mathscr{N}$ un langage, avec $\mathscr{N} = \{n\}$. On peut construire une implémentation des names qui correspond à représenter tous les names par un mot de la bonne longueur constitué unniquement de 0, de la manière suivante :*
*Soit $l \in \mathbb{N}$ : Alors la fonction $\big((n,f) \mapsto (0)^{f(l)}\big) \in \mathit{ImplNames}(\mathbf{\Lambda}, l)$ : c'est une interprétation des names pour $l$.*

On va maintenant définir une $\sigma$-algèbre sur l'ensemble ImplNames$(\mathbf{\Lambda}, l)$ :

**Definition 30.** *Soit $\mathbf{\Lambda} = \Sigma, \mathscr{V}, \mathscr{N}$.*
- *Soit $X \subseteq \mathscr{N} \times (\mathbb{N} \to \mathbb{N})$ un ensemble fini. Soit $l \in \mathbb{N}$. On définit*

$$\mathit{ImplNames}_X(\mathbf{\Lambda}, l) = \{\Theta : X \to (\{0,1\}^\star) \mid \forall (n,f) \in X, |\Theta(n,f)| = f(l)\}$$

- *Pour tout $X \subseteq \mathscr{N} \times (\mathbb{N} \to \mathbb{N})$ un ensemble fini, $l \in \mathbb{N}$, et $\Theta \in \mathit{ImplNames}_X(\mathbf{\Lambda}, l)$, on définit $(\uparrow^{name}\Theta)_X^{l,\mathbf{\Lambda}} = \{g \in \mathit{ImplNames}(\mathbf{\Lambda}, l) \mid \forall n \in X, \Theta(x) = g(x)\}$.*
- *On définit $AlgNames_l^{\mathbf{\Lambda}}$ comme la plus petite $\sigma$-algèbre qui contient les $(\uparrow^{name}\Theta)_X^{l,\mathbf{\Lambda}}$ pour tout $l \in \mathbb{N}$, $X \subseteq \mathscr{N} \times (\mathbb{N} \to \mathbb{N})$ un ensemble fini, et $\Theta \in \mathit{ImplNames}_X(\mathbf{\Lambda}, l)$.*

**Exemple 30.** *Soit $\mathbf{\Lambda} = (\Sigma, \mathscr{V}, \mathscr{N})$ un langage, et soient $n$ et $m$ deux éléments de $\mathscr{N}$.Soit $l$ un paramètre de sécurité. On s'intéresse à l'ensemble des implémentations des names tels que l'inmplémentation de $n$ et l'implémentation de $m$ sont identiques. On veut montrer qu'il s'agit d'un élément de la $\sigma$-algèbre $AlgNames_l^{\mathbf{\Lambda}}$.*
*On pose $X = \{(n, (p \mapsto p)), (m, (p \mapsto p))\}$, et pour tout $s \in \{0,1\}^l$, on pose*

$$\begin{aligned} \Theta_s : & X \to \{0,1\}^\star \\ & (.) \to s \end{aligned}$$

*qui est un élément de $\mathit{ImplNames}_X(\mathbf{\Lambda}, l)$.*
*On voit que l'ensemble des implémentations telles que $n^{(p \mapsto p)}$ et $m^{(p \mapsto p)}$ sont égaux peut être exprimé de la manière suivante :*

$$\{f \in \mathit{ImplNames}(\mathbf{\Lambda}, l) \mid f(n, (x \mapsto x)) = f(m, (x \mapsto x))\} = \bigcup_{s \in \{0,1\}^l} (\uparrow^{name}\Theta_s)_X^{l,\mathbf{\Lambda}}.$$

*Puisque cet ensemble est union dénombrable d'éléments générateurs de $AlgNames_l^{\mathbf{\Lambda}}$, on en déduit que c'est un élément de $AlgNames_l^{\mathbf{\Lambda}}$.*

**Definition 31.** *Soit $\mathbf{\Lambda}$ un langage de termes (implémentable), $l \in \mathbb{N}$ un paramètre de sécurité. Une interpretation des names de $\mathbf{\Lambda}$ pour le paramètre de sécurité $l$ est une mesure de probabilité*

$$\mathscr{D}_{names} : AlgNames_l^{\mathbf{\Lambda}} \to [0,1]$$

On remarque que pour définir une mesure de probabilité $\mathscr{D}_{names}$ sur $AlgNames_{\mathbf{\Lambda}}^l$, il est suffisant de définir sa valeur sur les unions disjointes d'éléments qui la génèrent.

**Lemme 6.** *Soit $\mathbf{\Lambda} = \Sigma, \mathscr{V}, \mathscr{N}$. Soit $l \in \mathbb{N}$. On définit :*

$$S \stackrel{def}{=} \{A \mid \exists X \subseteq \mathscr{N} \times (\mathbb{N} \to \mathbb{N}) \text{ un ensemble fini }, \exists \Theta \in \mathit{ImplNames}_X(\mathbf{\Lambda}, l),\ A = (\uparrow^{name}\Theta)_X^{l,\mathbf{\Lambda}}\}$$

*Soit $\mathscr{D} : S \to [0,1]$ qui vérifie :*
- *$\forall Y \in S, \mathscr{D}(Y) \geq 0$, $\mathscr{D}(\emptyset) = 0$*
- *$\mathscr{D}(\bigcup_{i \in \mathbb{N}} A_i) = \sum_{i \in \mathbb{N}} \mathscr{D}(A_i)$ si les $A_i \in semiring$ sont deux à deux disjoints, et $\bigcup_{i \in \mathbb{N}} A_i \in S$.*
- *$\mathscr{D}(\mathit{ImplNames}(\mathbf{\Lambda}, l)) = 1$*

*Alors il existe une unnique mesure de probabilité sur $AlgNames_{\mathbf{\Lambda}}^l$ qui coincide avec $\mathscr{D}$ sur son domaine de définition.*

*Démonstration.* Soit $\mathbf{\Lambda} = \Sigma, \mathscr{V}, \mathscr{N}$ un langage. Soit $l \in \mathbb{N}$. Il suffit de montrer que l'ensemble S est un semi-anneau, c'est-à-dire qu'il vérifie :
- $\emptyset \in S$
- Pour $A, B \in S$, on a $A \cap B \in S$
- Pour $A, B \in S$, on a : $A \setminus B$ qui peut s'écrire comme union finie disjointe d'éléments de $S$ : il existe $m \in \mathbb{N}$, et une famille $(K_i)_{i \leq m}$ deux à deux disjoints tels que $A \setminus B = \bigsqcup_{1 \leq i \leq m} K_i$

Et le théorème d'extension de Caratheodory entraîne le résultat.

□

**Exemple 31.** *On considère le langage **Strings**. Soit $l$ un paramètre de sécurité. On peut définir de la manière suivante une distribution sur les names pour laquelle, avec probabilité 1, l'implémentation d'un name $n^f$ est égale à $(0)^{f(l)}$ :*
*On définit $\mathscr{E}$ de la manière suivante : pour toute famille $(X_i, \Theta_i)$, où $X_i \subseteq \mathscr{N} \times (\mathbb{N} \to \mathbb{N})$ un ensemble fini et $\Theta_i \in$ ImplNames$_X(\mathbf{\Lambda}, l)$ tel que les $(\uparrow^{name}\Theta_i)^{l,\mathbf{\Lambda}}_{X_i}$ sont disjoints deux à deux :*

$$\mathscr{E}(\bigsqcup_{1\leq i\leq m}((\uparrow^{name}\Theta_i)^{l,\mathbf{\Lambda}}_{X_i})) = \sum_{1\leq i\leq m}\begin{cases} 1 \text{ si } \forall (n,f)\in X_i, \Theta_i(n,f) = (0)^{f(l)} \\ 0 \text{ sinon} \end{cases}$$

**Exemple 32.** *Soit $\mathbf{\Lambda} = \Sigma, \mathscr{V}, \mathscr{N}$ un langage. Soit $l$ un paramètre de sécurité. On peut définir de la manière suivante une distribution sur les names tels que : avec probabilité $\frac{1}{2}$, tous les names sont égaux au mot formé uniquement de $0$ de la bonne longueur, et avec probabilité $\frac{1}{2}$ ils sont tous égaux au mot formé uniquement de $1$ : On définit $\mathscr{E}$ de la manière suivante : pour toute famille $(X_i, \Theta_i)$, où $X_i \subseteq \mathscr{N} \times (\mathbb{N} \to \mathbb{N})$ un ensemble fini et $\Theta_i \in$ ImplNames$_X(\mathbf{\Lambda}, l)$ tel que les $(\uparrow^{name}\Theta_i)^{l,\mathbf{\Lambda}}_{X_i}$ sont disjoints deux à deux :*

$$\mathscr{E}(\bigsqcup_{1\leq i\leq m}((\uparrow^{name}\Theta_i)^{l,\mathbf{\Lambda}}_{X_i})) = \sum_{1\leq i\leq m}\begin{cases} \frac{1}{2} \text{ si } \forall (n,f)\in X_i,\ \Theta_i(n,f) = (0)^{f(l)} \\ \frac{1}{2} \text{ si } \forall (n,f)\in X_i,\ \Theta_i(n,f) = (1)^{f(l)} \\ 0 \text{ sinon} \end{cases}$$

**Exemple 33.** *Soit $l$ un paramètre de sécurité. On peut définir de la manière suivante une distribution uniforme sur les names (c'est-à-dire que, pour un name donné, touts les mots sur l'alphabet $\{0,1\}$ de la bonne longueur ont la même probabilité d'être choisi : On définit $\mathscr{E}$ de la manière suivante : pour toute famille $(X_i, \Theta_i)$, où $X_i \subseteq \mathscr{N} \times (\mathbb{N} \to \mathbb{N})$ un ensemble fini et $\Theta_i \in$ ImplNames$_X(\mathbf{\Lambda}, l)$ tel que les $(\uparrow^{name}\Theta_i)^{l,\mathbf{\Lambda}}_{X_i}$ sont disjoints deux à deux :*

$$\mathscr{E}(\bigsqcup_{1\leq i\leq m}((\uparrow^{name}\Theta_i)^{l,\mathbf{\Lambda}}_{X_i})) = \sum_{1\leq i\leq m}\prod_{(n,f)\in X_i}\frac{1}{2^{f(l)}}$$

**Definition 32.** *Soit $\mathscr{N}$ un ensemble de names, et $\mathscr{D}$ une interprétations de ces names. On dit que $\mathscr{D}$ est invariante par substitution des names, si : $\forall A \in AlgNames^{\mathbf{\Lambda}}_{,}$ $\forall \gamma$ une substitution sur l'ensemble $\mathscr{N}$,*

$$\mathscr{D}(A) = \mathscr{D}(\gamma(A))$$

*, où $\gamma(A) = \Theta \mid \exists \Theta' \in A$ tel que $\forall n \in \mathscr{N}, \forall f : \mathbb{N} \to \mathbb{N}, \Theta'(n,f) = \Theta(\gamma(n), f)$*

## D.4 preuve des lemmes concernant la fonction de hashage

On cherche à prouver le lemme suivant :

**Lemme 7.** *Soit $\mathcal{C}$ une contrainte, soit $M_1, ..., M_r$ les termes tels que $\mathbf{hash}(M_i)$ est un sous-terme d'un terme apparaissant dans $\mathcal{C}$. Alors, pour $n_1, .., n_r$ des fresh names :*

$$\begin{aligned}\mathscr{M}(\llbracket\mathcal{C}\rrbracket_l) \leq \sum_{I_1,..,I_k \text{ partition de } \{1,...,r\}} \mathscr{M}(\llbracket\mathcal{C}\ \{\forall j, n_{\bar{j}}/\mathbf{hash}(M_j)\} \\ \wedge M_{i_1}\{\forall j, n_{\bar{j}}/\mathbf{hash}(M_j)\} = M_{i_2}\{\forall j, n_{\bar{j}}/\mathbf{hash}(M_j)\}\forall i_1,\ i_2 \text{ tel que } \bar{i}_1 = \bar{i}_2 \\ \wedge \text{not}\left(M_{i_1}\{\forall j, n_{\bar{j}}/\mathbf{hash}(M_j)\} = M_{i_2}\{\forall j, n_{\bar{j}}/\mathbf{hash}(M_j)\}\right)\ \forall i_1,\ i_2 \text{ tel que } \bar{i}_1 \neq \bar{i}_2\rrbracket_l)\end{aligned}$$

*où, si $I_1, ..., I_k$ est une partition de $\{1, ..., r\}$, on note $\bar{i} = j$ pour $i \in I_j$.*

*Démonstration.* On pose $q = \mathrm{f}(p)$. On pose $d$ le maximal pour les $i$ du nombre d'occurence de $\mathbf{hash}$ appliqué à un terme de longueur $p$ dans la contrainte . On note $N_1, ...N_d$ tous les termes de longueur $p$ tel que $\mathbf{hash}N_j^i$ apparaît dans la contrainte.
On remarque que $d \leq r$.

$$\begin{aligned}
D &= \mathscr{M}_l\left([\![\mathcal{C}]\!]_l\right) \\
&= \sum_{\beta_1,...,\beta_d \in \{0,1\}^p} \mathscr{M}_l\left([\![\mathcal{C} \wedge \forall j \in \{1,...,d\}, N_j = \beta_j]\!]_l\right) \\
&= \sum_{I \text{ une partition de } \{1,..d\}} \sum_{\beta_1,...,\beta_{\#(I)} \in \{0,1\}^q} \mathscr{M}_l\left([\![\mathcal{C} \wedge \forall j \in \{1,...,d\}, N_j = \beta_{\overline{j}}]\!]_l\right) \\
&= \sum_{I \text{ une partition de } \{1,..d\}} \sum_{\beta_1,...,\beta_{\#(I)} \in \{0,1\}^p} \sum_{\alpha_1,...,\alpha_{\#(I)} \in \{0,1\}^q} \mathscr{M}_l\left([\![\mathcal{C} \wedge \forall j, N_j = \beta_{\overline{j}} \wedge \mathbf{hash}(\beta_j) = \alpha_{\overline{j}}]\!]_l\right)
\end{aligned}$$

Pour tout terme $L$, étant donné une partition I de cardinal $k$, et $\beta_1, ..., \beta_k$ et, $\alpha_1, .., \alpha_k$, on note $\overline{M}$ le terme $L$ où on a remplacé toute occurence de $\mathbf{hash}(N_j)$ par $\alpha_{\overline{j}}$, et on utilise la même notation pour les contraintes. Le point important est que l'implémentations des termes obtenus ne dépend plus de l'implémentation de $\mathbf{hash}$.

On a alors :

$$D = \sum_{I \text{ une partition de } \{1,..d\}} \sum_{\beta_1,...,\beta_{\#(I)} \in \{0,1\}^p} \sum_{\alpha_1,...,\alpha_k \in \{0,1\}^q} [\![\overline{\mathcal{C}_i} \wedge \forall j, \overline{N_j^i} = \beta_j \wedge \mathbf{hash}(\beta_j) = \alpha_{\overline{j}}]\!]_l$$

Comme l'implémentation de $\mathbf{hash}$ est indépendante des implémentations des autres constructeurs (et des names), et que les $\beta_j$ sont des constantes (ne dépendent pas de l'implémentation choisie), on peut écrire :

$$D = \sum_{I \text{ une partition de } \{1,..d\}} \sum_{\left(\beta_1,...,\beta_{\#(I)} \in \{0,1\}^p\right)} \sum_{\alpha_1,...,\alpha_k \in \{0,1\}^q} \mathscr{M}_l\left([\![\forall j \mathbf{hash}(\beta_j) = \alpha_{\overline{j}}]\!]_l\right) \cdot \mathscr{M}_l\left([\![\overline{\mathcal{C}_i} \wedge \forall j, \overline{N_j^i} = \beta_j]\!]_l\right)$$

On uttilise maintenant le fait que l'interprétation de $\mathbf{hash}$ est unniforme :

$$D = \sum_{I \text{ une partition de } \{1,..d\}} \sum_{\left(\beta_1,...,\beta_{\#(I)} \in \{0,1\}^p\right)} \sum_{\alpha_1,...,\alpha_{\#(I)} \in \{0,1\}^q} \frac{1}{2^{\#(I)\cdot q}} \cdot \mathscr{M}_l\left([\![\overline{\mathcal{C}_i} \wedge \forall j, \overline{N_j^i} = \beta_j]\!]_l\right)$$

On conclut en utilisant le fait que l'interprétation des names est unniforme.

□

Les deux autres lemmes sont des corollaires de celui-ci.

## D.5 Sémantique du $\pi$-calcul appliqué avec égalité probabiliste

### D.5.1 Congruence structurelle

La figure 6 définit la relation de congruence structurelle pour le $\pi$-calcul avec égalité probabiliste.

Le lemme suivant indique qu'on peut toujours écrire un processus comme un processus sans contrainte en parralèle avec une contrainte, l'ensemble étant soumis à des restrictions :

**Lemme 8.** *Soit $\mathbf{\Lambda}$ un langage, $P \in \pi^{\mathcal{C}}(\mathbf{\Lambda})$. Alors il existe $\mathcal{C}$ une contrainte sur $\mathbf{\Lambda}$, $A_1$ un processus étendu sans contrainte, et $\overline{u}$ une liste de names, tels que :*

$$P \equiv \nu \overline{u} \cdot (\mathcal{C} \mid A_1)$$

– : les relations usuelles du $\pi$-calcul appliqué :

$$
\begin{array}{rll}
B_1 \equiv (B_1 \mid 0) & & \text{PAR-0} \\
(B_1 \mid B_2) \equiv (B_2 \mid B_1) & & \text{PAR-C} \\
(B_1 \mid (B_2 \mid B_3)) \equiv ((B_1 \mid B_2) \mid B_3) & & \text{PAR-A} \\
!\,(B_1) \equiv (B_1 \mid !\,(B_1)) & & \text{REPL} \\
\nu n^{\lambda} \cdot 0 \equiv 0 & & \text{NEW-0} \\
\nu u^{\lambda} \cdot \nu v^{\lambda_2} \cdot B_1 \equiv \nu v^{\lambda_2} \cdot \nu u^{\lambda} \cdot B_1 & & \text{NEW-C} \\
\left(B_1 \mid \nu u^{\lambda} \cdot B_2\right) \equiv \nu u^{\lambda} \cdot (B_1 \mid B_2) & \text{si } u^{\lambda} \notin fv(B_1) \cup fn(B_1)) & \text{NEW-PAR} \\
\nu x^{\lambda} \cdot \left\{\{M/x^{\lambda}\}\right\} \equiv 0 & \text{si } \forall l, \mid M \mid_l = \lambda(l) & \text{ALIAS} \\
\left(B_1 \mid \left\{\{M/x^{\lambda}\}\right\}\right) \equiv \left(A\{M/x^{\lambda}\} \mid \left\{\{M/x^{\lambda}\}\right\}\right) & & \text{SUBST}
\end{array}
$$

– Règle spécifique aux contraintes :

$$(\mathcal{C}_1 \mid \mathcal{C}_2) \equiv \mathcal{C}_1 \wedge \mathcal{C}_2$$

FIGURE 6 – Règles pour la congruence structurelle $\equiv$

#### D.5.2 Contexte d'évaluation

On définit ici la notion de contexte d'évaluation :

$$\mathscr{C}[.] \;::=\; [.] \;\Big|\; (\mathscr{C}[.] \mid A_1) \;\Big|\; \nu x^{\lambda} \cdot \mathscr{C}[.] \;\Big|\; \nu n^{\lambda} \cdot \mathscr{C}[.]$$

#### D.5.3 Sémantique labellée

La figure 7 définit le relation de sémantique labellée pour le $\pi$-calcul avec égalité probabiliste.

## E $\pi$-calcul appliqué avec égalité probabiliste et phases rapides

#### E.0.4 Exemples de processus bien formé

**Exemple 34.**
– $(end \cdot 0)$ *n'est pas un processus bien formé.*
– $(begin\,(w_1) \cdot \ if\ (M = N)\ then\ end \cdot 0\ else\ 0)$ *n'est pas un processus bien formé : si on prends le chemin d'exécution qui passe par la branche* $else$*, on a un* $begin\,(\cdot) \cdot$ *qui n'est pas suivi d'un* $end$*·*
– $begin\,(w_1) \cdot begin\,(w_1) \cdot end \cdot end \cdot 0$ *n'est pas bien formé, parce que les* $begin$ *et les* $end$ *sont emboîtés les uns dans les autres.*
– $(begin\,(w_1) \cdot \ if\ (M = N)\ then\ end \cdot 0\ else\ \overline{p}\,(M) \cdot end \cdot 0)$ *est bien formé*

$$\frac{\dfrac{\dfrac{\tau\,\dfrac{B_1 \to \mu}{B_1 \to^{\tau\ l} \mu}}{(\mathcal{C} \mid \overline{p}\,(x) \cdot P) \to^{\overline{p}(x)\ l} \delta_{(\mathcal{C}|P)}}\ \text{out-atom}}{(\mathcal{C} \mid p\,(x) \cdot P) \to^{p(M)\ l} \delta_{(\mathcal{C}|P\{M/x\})}}\ \text{in} \qquad \downarrow M = M_{\phi(A_1)} \text{ est un “ground terme”} \qquad \downarrow N = N_{\phi(A_1)} \text{ est un “ground terme”}}{(\mathcal{C} \mid A_1) \to^{\text{test}(M,N)\ l} [\downarrow M =\downarrow N \mid \mathcal{C}]_l \cdot \delta_{(\mathcal{C} \wedge (\downarrow M = \downarrow N) | A_1)} + (1 - [\downarrow M =\downarrow N \mid \mathcal{C}]_l) \cdot \delta_{(\mathcal{C} \wedge \text{not}(\downarrow M = \downarrow N) | A_1)}}\ \text{test}$$

$$\frac{\dfrac{B_1 \to^{\overline{a}(u)\ l} \mu \qquad a \neq u}{\nu u \cdot B_1 \to^{\nu u \cdot \overline{a}(u)\ l} \mu}\ \text{open-atom} \qquad B_1 \to^{\alpha\ l} \mu \qquad bn(\alpha) \cap fn(A_1) = \emptyset = bv(\alpha) \cap fv(A_1) \qquad A_1 \text{ processus sans contrainte}}{(B_1 \mid A_1) \to^{\alpha\ l} (\mu \mid A_1)}\ \text{PAR}$$

$$\frac{B_1 \to^{\alpha\ l} \mu \qquad x\text{doesn't occur in } \alpha}{\nu x \cdot B_1 \to^{\alpha\ l} \nu x \cdot \mu}\ \text{Scope} \qquad \frac{B_1 \to^{\alpha\ l} \mu \qquad B_1 \equiv B_2}{B_2 \to^{\alpha\ l} \mu}$$

$M_{\phi(A_1)}$ représente le terme $M$ auquel on a appliqué la substitution impliquée par les substitutions actives du processus $A_1$.
La règle out-atom modélise l'émission d'un message. La règle in modélise la réception d'un message. La règle test modélise les fait que l'adversaire peut tester l'égalité sur les termes qu'il connaît. Les règles Par, Struct et Scope sont des règles de passage au contexte.

FIGURE 7 – Sémantique labellée pour le $\pi$-calcul appliqué avec égalité probabiliste

$$B_1 \equiv (B_1 \mid 0)$$
$$(B_1 \mid B_2) \equiv (B_2 \mid B_1)$$
$$(B_1 \mid (B_2 \mid B_3)) \equiv ((B_1 \mid B_2) \mid B_3)$$
$$!\,(B_1) \equiv (B_1 \mid !\,(B_1))$$
$$\nu n^{\lambda} \cdot 0 \equiv 0$$
$$\nu u^{\lambda} \cdot \nu v^{\lambda_2} \cdot B_1 \equiv \nu v^{\lambda_2} \cdot \nu u^{\lambda} \cdot B_1$$
$$\left(B_1 \mid \nu u^{\lambda} \cdot B_2\right) \equiv \nu u^{\lambda} \cdot (B_1 \mid B_2) \quad \text{si } u^{\lambda} \notin fv(B_1) \cup fn(B_1))$$
$$\nu x^{\lambda} \cdot \left\{\{M/x^{\lambda}\}\right\} \equiv 0 \quad \text{si } \forall l, \mid M \mid_l = \lambda(l)$$
$$\left(B_1 \mid \left\{\{M/x^{\lambda}\}\right\}\right) \equiv \left(B_1\{M/x^{\lambda}\} \mid \left\{\{M/x^{\lambda}\}\right\}\right)$$

(a) Règles similaires à celles du $\pi$-calcul appliqué

$$(\mathcal{C}_1 \mid \mathcal{C}_2) \equiv \mathcal{C}_1 \wedge \mathcal{C}_2$$

(b) Règle spécifique aux contraintes

$$(\text{wait}^{w_1}\,(X) \cdot P \mid \text{wait}^{w_1}\,(X) \cdot Q) \equiv \text{wait}^{w_1}\,(X) \cdot (P \mid Q)$$
$$\text{wait}^{w_1}\,(X) \cdot \overline{p}\,(M) \cdot P \equiv \overline{p}\,(M) \cdot \text{wait}^{w_1}\,(X) \cdot P \text{ si } X \cap Var(M) = \emptyset \wedge p \notin X$$
$$\text{wait}^{w_1}\,(X) \cdot p\,(y) \cdot P \equiv p\,(y) \cdot \text{wait}^{w_1}\,(X) \cdot P \text{ si } y \notin X \wedge p \notin X$$
$$\text{wait}^{w_1}\,(X) \cdot \text{ if } (M = N) \text{ then } P \text{ else } Q \equiv \text{ if } (M = N) \text{ then wait}^{w_1}\,(X) \cdot P \text{ else wait}^{w_1}\,(X) \cdot Q \text{ si } (Var(M) \cup Var(N)) \cap X = \emptyset$$
$$\text{wait}^{w_1}\,(X) \cdot \text{wait}^{w_1}\,(Y) \cdot P \equiv \text{wait}^{w_1}\,((X \cup Y)) \cdot P$$
$$\text{wait}^{w_1}\,(X) \cdot 0 \equiv 0$$

(c) Règles spécifiques au wait

FIGURE 8 – Congruences structurelles pour le $\pi$-calcul appliqué avec contraintes et phase rapides

$$\left(\mathcal{C} \mid \left(\overline{p}\left(x^{\lambda}\right) \cdot P \mid p\left(x^{\lambda}\right) \cdot Q\right)\right), \top \rightarrow \delta_{(\mathcal{C}|(P|Q)),\top}$$

$$\left(\mathcal{C} \mid \left(\overline{\text{fast}(p,\cdot)}\,(x) \cdot P \mid \text{fast}(p,\cdot)\,(x) \cdot Q\right)\right), \top \rightarrow \delta_{(\mathcal{C}|(P|Q)),\top}$$

$$\left(\mathcal{C} \mid \left(\overline{\text{fast}(p,\cdot)}\,(x) \cdot P \mid \text{fast}(p,w_1)\,(x) \cdot Q\right)\right), \text{lock}\,(w_1^n) \rightarrow \delta_{(\mathcal{C}|(P|Q)),\text{lock}\left(w_1^n\right)}$$

$$\left(\mathcal{C} \mid \left(\text{fast}(p,w_2)\,(y) \cdot P \mid \overline{\text{fast}(p,\cdot)}\,(x) \cdot Q\right)\right), \text{lock}\,(w_1^n) \rightarrow \delta_{(\mathcal{C}|(\text{wait}^{w_1}(\{x\})\cdot P|Q)),\text{lock}\left(w_1^n\right)}$$

$$(\text{ if } (M=N) \text{ then } P \text{ else } Q \mid \mathcal{C}),\ \mathbf{l}_1 \rightarrow [M=N \mid \mathcal{C}]_l \cdot \delta_{(P|\mathcal{C})\wedge(M=N),\mathbf{l}_1} + (1-[M=N \mid \mathcal{C}]_l) \cdot \delta_{(Q|\mathcal{C})\wedge\text{not}(M)=N,\mathbf{l}_1}$$

$$(\mathcal{C} \mid \text{begin}\,(w_1) \cdot P), \text{lock}\,(w_1^n) \rightarrow \delta_{(\mathcal{C}|P),\text{lock}\left(w_1^{n+1}\right)}$$

$$(\mathcal{C} \mid (\text{end} \cdot P \mid \text{wait}^{w_1}(X_1) \cdot Q_n) \mid \cdots \mid \text{wait}^{w_1}(X_n) \cdot Q_n),\ \text{lock}\,(w_1^n) \rightarrow \delta_{(\mathcal{C}|(P|Q_1|\cdots|Q_n)),\text{lock}\left(w_1^{n-1}\right)}\ ,\ \text{avec } n \geq 1$$

$$\frac{B_1 \rightarrow \mu}{\mathscr{C}[B_1] \rightarrow \mu_{\mathscr{C}[]}}$$

FIGURE 9 – Sémantique non labellée pour le $\pi$-calcul appliqué avec égalité probabiliste et phases rapides

### E.0.5 Congruence structurelle pour le $\pi$-calculappliqué avec égalité probabiliste et phases rapides

### E.0.6 Sémantique non labellée

Les états du LTS probabiliste pour la sémantique non labellée sont les couples de la forme $(B_1, \mathbf{l}_1)$, où $\mathbf{l}_1$ est un lock sur les localisations, engendrés par la grammaire suivante :

$$\mathbf{l}_1 \ ::= \ \top \mid \text{lock}\,(w_1) \ \text{avec } w_1 \in Loc$$

:

### E.0.7 Sémantique labellée

Les états du LTS probabiliste pour la sémantique labbellée sont de la forme $B_1, V, \mathbf{l}_1$, où $B_1$ est un processus étndu avec contraintes, $\mathbf{l}_1$ est un lock sur les localisations, et $V$ est une suite(ou un ensemble) de variables $x_1^{w_1}, ..., x_n{}^{w_n}$ qui correspondent à des termes qui ne lui sont accessibles que si il est situé dans la sphère correspondant à ces locations.

$$V \ ::= \ \emptyset \ \Big| \ \{x\}^{w_1}$$

$$\frac{B_1, \top \to \delta_{B_2, \mathrm{lock}\left(w_1^1\right)} \qquad n \geq 1}{B_1, \emptyset, \top \to^{\tau} \delta_{B_2, \emptyset, \mathrm{lock}\left(w_1^1\right)}} \qquad \frac{B_1, \mathrm{lock}\left(w_1^n\right) \to \delta_{B_2, \mathrm{lock}\left(w_1^{n+1}\right)} \qquad n \geq 1}{B_1, V, \mathrm{lock}\left(w_1^n\right) \to^{\tau} \delta_{B_2, V, \mathrm{lock}\left(w_1^{n+1}\right)}}$$

$$\frac{B_1, \mathrm{lock}\left(w_1^n\right) \to \delta_{B_2, \mathrm{lock}\left(w_1^{n-1}\right)} \qquad n \geq 2}{B_1, V, \mathrm{lock}\left(w_1^n\right) \to^{\tau} \delta_{B_2, V, \mathrm{lock}\left(w_1^{n-1}\right)}}$$

$$\frac{B_1, \mathrm{lock}\left(w_1^1\right) \to \delta_{B_2, \top} \qquad n \geq 2}{B_1, V, \mathrm{lock}\left(w_1^1\right) \to^{\tau} \delta_{B_2, \emptyset, \top}} \qquad \frac{B_1, \mathrm{l}_1 \to \mathscr{D}, \mathrm{l}_1}{B_1, V, \mathrm{l}_1 \to^{\tau} \mathscr{D}, V, \mathrm{l}_1}$$

(a) $\tau$-actions : actions internes au processus

$$\left(\mathcal{C} \mid \overline{p}\left(x\right) \cdot P\right), \emptyset, \top \to^{\overline{p}(x)} \left(\mathcal{C} \mid P\right), \emptyset, \top \qquad \text{out-atom}$$

$$\frac{B_1, \emptyset, \top \to^{\overline{a}(u)} \mu \qquad a \neq u}{\nu u \cdot B_1, \emptyset, \top \to^{\nu u \cdot \overline{a}(u)} \mu} \qquad \text{open-atom}$$

$$\left(\mathcal{C} \mid p\left(x\right) \cdot P\right), \emptyset, \top \to^{p(M), \emptyset} \delta_{(\mathcal{C} \mid P\{M/x\}),, \emptyset, \top} \qquad \text{in}$$

(b) Règles similaires au $\pi$-calcul appliqué, correspondant aux communications normales

$$\left(\mathcal{C} \mid \overline{\mathrm{fast}(p, w_2)}\left(x\right) \cdot P\right), V, \mathrm{lock}\left(w_1^n\right) \to^{(\overline{\mathrm{fast}(p, w_2)}(x))} \left(\mathcal{C} \mid P\right), V, \mathrm{lock}\left(w_1^n\right) \quad \text{avec } n \geq 1$$

$$\frac{B_1, V, \mathrm{lock}\left(w_1^n\right) \to^{\overline{\mathrm{fast}(p, \cdot)}(x)} \delta_{B_2, V, \mathrm{lock}(w_1)}}{\nu x \cdot B_1, V, \mathrm{l}_1 \to^{\nu x \cdot \overline{\mathrm{fast}(p, \cdot)}(x)} \delta_{B_2, V \cup \{x\}^{w_1}, \mathrm{lock}\left(w_1^n\right)}} \text{ open-atom-fast}$$

$$\frac{\{w_2 \in Loc \mid \exists x \in Var(M) \text{ tel que } x^{w_2} \in V\} \subseteq S \qquad n \geq 1}{\left(\mathcal{C} \mid \mathrm{fast}(p, w_1)\left(x\right) \cdot P\right), V, \mathrm{lock}\left(\mathrm{l}_1^n\right) \to^{\mathrm{fast}(p,x)(M), S} \delta_{(\mathcal{C} \mid P\{M/x\}),, V, \mathrm{lock}\left(\mathrm{l}_1^n\right)}} \text{ fast-in-1}$$

$$\frac{\{w_2 \in Loc \mid \exists x \in Var(M) \text{ tel que } x^{w_2} \in V\} \subseteq S \qquad n \geq 1 \qquad w_3 \neq w_1}{\left(\mathcal{C} \mid \mathrm{fast}(p, w_3)\left(x\right) \cdot P\right), V, \mathrm{lock}\left(\mathrm{l}_1^n\right) \to^{\mathrm{fast}(p,w_3)(M), S} \delta_{(\mathcal{C} \mid \mathrm{wait}^{w_1}(x) \cdot P\{M/x\}),, V, \mathrm{lock}\left(\mathrm{l}_1^n\right)}} \text{ fast-in-2}$$

(c) Communication pendant les phases rapides

$$\frac{M_{\phi(A_1)},\ N_{\phi(A_1)} \text{ sont des “ground terme} \qquad \{w_2 \in Loc \mid \exists x \in Var(M) \cup Var(N) \text{ tel que } x^{w_2} \in V\} \subseteq S}{\left(\mathcal{C} \mid A_1\right), V, \mathrm{l}_1 \to^{\mathrm{test}_S(M,N)} \left[M_{\phi(A_1)} = N_{\phi(A_1)} \mid \mathcal{C}\right]_l \cdot \delta_{\left(\mathcal{C} \wedge M_{\phi(A_1)} = N_{\phi(A_1)} \mid A_1\right)} + \left(1 - \left[M_{\phi(A_1)} = N_{\phi(A_1)} \mid \mathcal{C}\right]_l\right) \cdot \delta_{\left(\mathcal{C} \wedge \mathrm{not}\left(\left(M_{\phi(A_1)} = N_{\phi(A_1)}\right)\right) \mid A_1\right.}}$$

(d) Règle correpondant aux tests effectués par l'adversaire

$$\frac{B_1, \Phi, w_1 \to^{\alpha} \mu \qquad bn(\alpha) \cap fn(A_1) = \emptyset = bv(\alpha) \cap fv(A_1)}{\left(B_1 \mid A_1\right), \Phi, w_1 \to^{\alpha} \left(\mu \mid A_1\right)} \text{ PAR}$$

$$\frac{B_1, \Phi, w_1 \to^{\alpha} \mu \qquad x\text{doesn't occur in } \alpha}{\nu x \cdot B_1, \Phi, w_1 \to^{\alpha} \nu x \cdot \mu} \text{ SCOPE}$$

$$\frac{B_1, \Phi, w_1 \to^{\alpha} \mu \qquad B_1 \equiv B_2}{B_2, \Phi, w_1 \to^{\alpha} \mu} \text{ STRUCT}$$

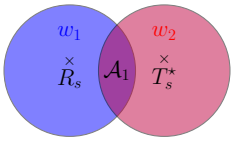
(e) Règles de passage au contexte

FIGURE 10 – Règles de la sémantique labellée pour le $\pi$-calcul appliqué avec égalité probabiliste et échange rapide

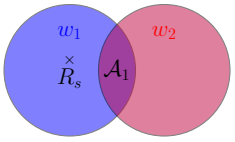
# F Attaques

**Attaque terroriste :** On considère la situation suivante : un tag $T^\star$, avec la complicité de l'adversaire, cherche à faire croire à un lecteur éloigné $R$, avec qui il partage la même clé secrète $s$, qu'il est proche de lui. Il y a une attaque seulement si l'adversaire ne peut s'identifier qu'avec la collaboration du tag, et pas s'identifier à nouveau par lui-meme en utilisant les informations que le tag lui a fourni précédemment.

On doit donc considérer trois phases successives : La phase 1 est une phase d'apprentissage. La phase 2 peut être représenté de la manière suivante :




La phase 3 peut être représenté de la manière suivante : $T^\star$ ne collabore plus avec l'adversaire, et l'adversaire cherche à nouveau à faire croire au lecteur $R$ qu'un tag correspondant est proche de $R$.




Il y a une attaque si il existe un tag corrompu $T^\star$, tel qu'il existe au moins un adversaire tel que la phase 2 réussit, et que la phase 3 échoue pour tous les adversaires.

# G Evolution probabiliste d'un processus

## G.1 Exécution du processus

- on note $B_1, \mathbf{l}_1 \rightarrow_\mu B_2, \mathbf{l}_2$, if $B_1, \mathbf{l}_1 \rightarrow \mu$ et $\mu(B_2, \mathbf{l}_2) > 0$.
- une execution de $A_1, \omega_1$ est une suite finie ou infinie $(A_1, \omega_1) \rightarrow_{\mu_1}, ...(A_n, \omega_n)$.... On note cet ensemble Exec $(A_1, \omega_1)$. On note cet ensemble Exec $(A_1, \omega_1)$. Si $e$ est une execution, on note $(\uparrow e)$ l'ensemble des exécutions dont $e$ est un préfixe.
- a labelled execution de $A_1, \Phi_1, V_1, \omega_1$ est une suite finie ou infinie $(A_1, \Phi_1, V_1, \omega_1) \rightarrow_{\mu_1}, ...(A_n, \Phi_n, V_n, \omega_n)$.... On note cet ensemble $\text{Exec}^{\text{labelled}}(A_1, \Phi_1, V_1, \omega_1)$

## G.2 Schedulers

- On note behave $(B_1, \mathbf{l}_1) = \{\mu \mid B_1, \mathbf{l}_1 \rightarrow \mu\}$, et behave$_l$ $(B_1, \Phi, V, \mathbf{l}_1) = \{(\alpha, \mu) \mid B_1, \Phi, V, \mathbf{l}_1 \rightarrow^\alpha \mu\}$
- On va définir la notion de scheduler pour la sémantique non labéllée : le but est de résoudre explicitement le non-déterminisme de la sémantique :
  - Un scheduler (pour la sémantique non labelled) est une fonction totale

$$\sigma : e \in \{\text{executions}\} \rightarrow \text{behave}\,(\text{last}\,(e)) \cup \{\perp\}$$

    Le cas où $\sigma(e) = \perp$ correspond au cas où le scheduler décide d'arrêter l'exécution (en particulier, c'est le cas si aucune action n'est possible à partir de cet état, c'est-à-dire si : behave (last $(e)$) $= \emptyset$). On remarque que le scheduler n'est pas probabiliste : par exemple si $B_1, \mathbf{l}_1 \rightarrow \delta_s$, et $B_2, \mathbf{l}_1 \rightarrow \delta_t$ le scheduler indique si $B_1 \mid B_2 \rightarrow \delta_t$, ou bien $B_1 \mid B_2 \rightarrow \delta_s$ (mais pas une combinaison probabiliste des deux).

  - Etant donné un scheduler $\sigma$, on définit $Exec^\sigma_{B_1,\mathbf{l}_1}$ l'ensemble des exécutions dirigées par les scheduler $\sigma$ et dont le premier état est $(B_1, \mathbf{l}_1)$

  - Etant donné $e = (B_1, \mathbf{l}_1) \rightarrow_{\mu_1}, ...(A_n, \omega_n) \in Exec^\sigma_{B_1,\mathbf{l}_1}$,pour tout paramètre de sécurité $l$, on note $P^\sigma_{B_1,\mathbf{l}_1}(l) = \prod_{1 \leq i \leq n} \mu_i(l)(A_i, \omega_i)$.
- De manière analogue, on définit la notion de stratégie pour la sémantique labbéllée : une stratégie (pour la sémantique labellée) est une fonction totale :

$$\mathcal{S} : e \in \{\text{labelled executions}\} \rightarrow \text{behave}_l\,(\text{last}\,(e)) \cup \{\perp\}$$

**Definition 33.** *Soit $\gamma$ un scheduler, $B_1$ un processus, $\mathbf{l}_1$ une spécifications sur les localisations.*

- *On définit $\sigma Field^\gamma_{B_1,\mathbf{l}_1}$ comme la plus petite $\sigma$ algèbre qui contient $(\uparrow e)$ pour toute $e \in Exec^{B_1,\mathbf{l}_1}_\gamma$.*
- *Pour tout paramètre de sécurité $l$, on définit une mesure de probabilités $\left(Prob^\gamma_{B_1,\mathbf{l}_1}\right)_l$ sur $\sigma Field^\gamma_{B_1}$ de la manière suivante : c'est l'unnique mesure de probabilité tel que $\left(Prob^\gamma_{B_1}\right)_l((\uparrow e)) = P^\gamma_{B_1,\mathbf{l}_1}(e)(l)$.*

Given a scheduler $\sigma$, a process $B_1$, a location $\mathbf{l}_1$, un ensemble de processus $H$, une borne sur le nombre d'étapes $n$, on définit

$$\text{Exec}^{(B_1,\mathbf{l}_1),n}_\sigma(H) = \left\{e \in \text{Exec}\,(B_1, \mathbf{l}_1, \sigma) \mid \text{last}\left(e^i\right) \in H \text{ for some } i \leq n\right\}$$

.

**Lemme 9.** *If $B_1, \mathbf{l}_1 \rightarrow \mu$, then*

$$\mathit{Prob}^{(B_1,\mathbf{l}_1),n}_\sigma(H) = \sum_{B_2,\mathbf{l}_2 \in Support(\mu)} \mu(B_2, \mathbf{l}_2) \times \mathit{Prob}^{(B_2,\mathbf{l}_2),n-1}_\sigma(H)$$

# H Preuve du théorème sur l'équivalence d'un adversaire modélisé comme un processus pour la sémantique non labellée, et un adversaire modélisé par une stratégie pour la sémantique labellée.

## H.1 Définitions préliminaires

- Modélisation de l'adversaire par un processus s'éxécutant en parrallèle :
  - Un adversaire est modélisé par une suite de processus (sans frame ni contraintes, ni opérateurs de déclenchements ou de fin d'échange rapide) $(Adv)_l$ (où $l$ représente le paramètre de sécurité). L'ensemble $S = \{w_1 \mid w_1 \text{ occurs in } P\}$ est l'ensemble des locations disponnibles pour l'adversaire.
  - Alors la probabilité que, étant donné un paramètre de sécurité $l$, un système $B_1$ intéragissant avec l'adversaire $Adv$, partant d'un état où le lock indiquant la présence où non d'un échange rapide est égal à $\mathbf{l}_1$ atteigne un état $e \in E$, où E est caractérisé par une formule $\phi$ (en $\leq n$) étapes est calculé comme :

$$max_{\sigma \text{ a scheduler}} Pr\{(B_1 \mid \text{Adv}_l), \mathbf{l}_1 \rightarrow^{\sigma}_{\leq n} E\}$$

- un adversaire est modélisé par une stratégie sur le lts :

  Soit $S \subseteq Loc$ un ensemble de locations. Etant donné un paramètre de sécurité $l$, un adversaire ayant accès aux locations $S$ est caractérisé par la donnée d'une stratégie de l'adversaire, c'est-à-dire

  $\mathcal{A}_S : e \in \{\text{labelled executions}\} \times \Phi \rightarrow \{(\alpha, \mu) \in \text{behave}(\text{last}(e)) \text{ tel que } \text{Loc}(\alpha) \in S\}$ tel que $\text{hide}(e) = \text{hide}(f) \Rightarrow A_S(e$

  Pour chaque état du lts, l'adversaire connaît seulement les information données par $\Phi$, et le chemin effectué pour ariver dans cet état, et il doit décider d'une action.

  La probabilité que, étant donné une stratégie, un paramètre de sécurité $l$, et une borne sur le nombre d'étapes pour éxécuter le processus $n$, on atteigne un ensemble d'état E en partant d'un état $s$, avec un historique $h$ est exprimée par :
  On décompose l'état de départ de la manière suivante :

$$s = B_1, \Phi, \{\text{Var restreintes localement}\}, \mathbf{l}_1$$

$$P_l(\mathcal{A}, s, E, h, n) = \sum_{t \in \text{Support}(\mu)} \mu(t) \cdot P_l(\mathcal{A}, t, E, s \rightarrow^{\alpha}_{\mu} :: h, n-1),$$
$$\text{si } (\alpha, \mu) = \mathcal{A}(s)$$

  Dans le cas particulier où le calcul part de l'état s, on cherche à calculer : $P_l(\mathcal{A}, s, E, [], n)$

**Definition 34.** *Soit **result** un name.*
*On s'intéresse à la probabilité que l'on atteigne un état où le protocol a ou échoué, ou réussi : ceci est modélisé par le fait que l'on est dans un état de la forme $\left(\overline{\textbf{result}}(r) \cdot 0 \mid P\right)$, avec $r \in \{0, 1\}$. On note donc*

$$F_r = \{B_1 \mid \exists B_2 \text{ tel que } B_1 \equiv \overline{\textbf{result}}(r) \cdot 0 \mid B_2\}$$

**Definition 35.** *Soit **result** un name.*

- *On veut définir $Process^{\textbf{result}}$ de manière à ne considérer que des processus qui ne transmettent jamais à l'adversaire le canal **result**. Pour cela, on spécifie qu'aucun terme présent dans ce processus ne peut contenir **result** : On définit donc :*

$$Process^{\textbf{result}} = \{B_1 \text{ processus étendu contraint } \mid B_1 \equiv B_2 \text{ et } \nexists M, (M \in \textit{Terms}(B_2) \wedge \textbf{result} \in \textit{Names}(M))\}$$

– *On définit* $Adv_L^{\textbf{result}}$ *comme l'ensemble des processus modélisant un adversaire qui n'utilisent pas le canal* ***result*** *par :*

$$Adv_L^{\textbf{result}} = \{\textit{Processus } P \textit{sans contraintes, sans frame, sans déclenchement de l'échange rapide, et sans ifcontrol} \mid loc(P) \subseteq L \wedge \textbf{result} \notin fn(P)\}$$

– *De manière analogue, on définit l'ensemble des stratégies sur le lts labellé modélisant un adversaire qui possède les localisations* $L$ *et qui n'utilise pas le canal* ***result*** *par :*

$$Strat_L^{\textbf{result}} = \{\mathcal{S} \in \{\textit{Stratégies}\} \mid \forall e,\ \textit{si } \mathcal{S}(e) = (\alpha, \mu),\ \textit{alors } \textbf{result} \notin fn(\alpha) \wedge Loc(\alpha) \subseteq L\}$$

## H.2 Théorème

**Théorème 3.** *Pour* $r \in \{0,1\}$*, si on définit* $E_r = \{(B_1, \Phi, V, \mathbf{l}_1) \mid B_1 \in F_r \wedge \Phi, V, \mathbf{l}_1$ *quelconques* $\}$*,alors pour tout paramètre de sécurité* $l$*, pour toute borne de sécurité* $n$*, pour tout processus étendu contraint* $B_1 \in Process^{\textbf{result}}$*, pour tout ensemble de localisations* $L$ *:*

$$(max_{\mathcal{S} \in Strat_L^{\textbf{result}}}(P_l(\mathcal{S}, (B_1,\ \textit{Empty}\ , \emptyset, \top), E_r, [], n)) = max_{\mathcal{A}_1 \in Adv_L^{\textbf{result}}}\{max_{\sigma\ a\ scheduler} Pr\{(B_1 \mid \mathcal{A}_1)\ , \top \rightarrow_{\leq n}^{\sigma} F_r\}\}$$

**Lemme 10** (Contexte ⇒ Stratégie)**.** *On s'intéresse à la probabilité que l'on atteigne un état où le protocol a ou échoué, ou réussi : ceci est modélisé par le fait que l'on est dans un état de la forme* $\left(\overline{\textbf{result}}\,(r) \cdot 0 \mid P\right)$*, avec* $r \in \{0,1\}$*. On note donc* $F_r = \{B_1 \mid \exists B_2$ *tel que* $B_1 \equiv \overline{\textbf{result}}\,(r) \cdot 0 \mid B_2\}$

*Alors il existe*

$$\Theta : Adv_L^{\textbf{result}} \times \{\textit{schedulers}\} \rightarrow Strat_L^{\textbf{result}}$$

*, tel que :* $\forall s = B_1, \Phi, V, \mathbf{l}_1$ *état du lts, tel que Loc(*$V$*)* $\subseteq L$ *et* $B_1 \in Process^{\textbf{result}}$*, pour* $r \in \{0,1\}$ *si on définit* $E_r = \{B_2, \mathbf{l}_2 \mid B_2 \in F_r \wedge \mathbf{l}_2$ *quelconque* $\}$*, alors pour tout* $n$*, pour tout* $\mathcal{A}_1 \in Adv_L^{\textbf{result}}$ *:*

$$Pr(\mathcal{A}_1 \mid B_1, \mathbf{l}_1 \rightarrow_{\sigma}^{\leq n} E_r) \leq Pr(s \rightarrow_{\Theta(\mathcal{A}_1,\sigma)}^{\leq n} \textit{Proj}(E_r))$$

*, où Proj*$(E) = \{$*états du lts*$(B_1, \Phi, V, \mathbf{l}_1) \mid (B_1, \mathbf{l}_1) \in E\}$

*Démonstration.* L'idée générale est la suivante : donner un scheduler, revient à donner un ensemble d'exécution valables pour ce scheduler, qui doit vérifier les deux conditions suivantes : être stable par préfix, est tel que, si $e \rightarrow_\mu s$ est dans cet ensemble, alors il n'y a aucun élément de l'ensemble de la forme : $e \rightarrow_\gamma t$, et $\mu \neq \gamma$ (plus une troisième condition qui revient au fait que $\sigma$ ne voit pas les contraintes dans les processus). De même, donner une stratégie, c'est la même chose que de donner l'ensemble des exécutions labellées valable pour cette stratégie. La technique de preuve est donc la suivante : avoir une fonction $\phi$ de l'ensemble des exécutions non labellées de $B_1 \mid \mathcal{A}_1$ dans l'ensemble des exécutions labelléesde $B_1$, et ensuite, étant donné un scheduler $\sigma$, qui est caractérisé par un ensemble d'exécution $E$, définir la stratégie correspondante comme la stratégie caractérisée par l'ensemble d'exécutions labellées $\phi(E)$.

Pour les nécessités de la preuve, on ajoute des actions $s \rightarrow^\tau \delta_s$ pour tout état $s$ du lts.

On va d'abord démontrer le lemme suivant :

**Lemme 11.** *Soit* ***result*** *un name.*
*Soit un adversaire* $\mathcal{A}_1 \in Adv_L^{\textbf{result}}$*, et soit* $B_1 \in Process^{\textbf{result}}$*.*
*Alors il existe une fonction* ***injective*** *:*

$$\phi_{B_1} : \{\textit{executions non labellées de } (B_1 \mid \mathcal{A}_1)\} \rightarrow \{\textit{executions labbellees e de } B_1 \textit{ utilisant les locations } Loc(\mathcal{A}_1) \textit{ et tel que } \textbf{result} \notin fn(e)\}$$

*, qui vérifie les conditions suivantes :*

– *Pour tout $B_1$, la fonction $\phi_{B_1}$ respecte la structure des exécutions, c'est-à-dire que :* $\forall e, B_2, \mathbf{l}_1$*, il existe* $B_3, V, \Phi, \alpha, \rho$ *tels que* $\phi_{B_1}(e \rightarrow_\mu B_2, \mathbf{l}_1) = \left((\phi_{B_1}(e)) \rightarrow_\rho^\alpha B_3, V, \Phi, \mathbf{l}_1\right)$*, et que de plus :*
  – $\mu(B_2, \mathbf{l}_1) = \rho\left(B_3, V, \Phi, \mathbf{l}_1\right)$*, et il existe une processus* $\mathcal{A}_2$ *modélisant un adversaire tel que :* $B_3 \equiv \nu\overline{n} \cdot (B_2 \mid \mathcal{A}_2)$
  – $\mathcal{A}_2 \in Adv_L^{\textbf{result}}$ *et* $B_2 \in Process^{\textbf{result}}$
  – $\mathbf{l}_2 = lock\,(w_1) \Rightarrow \forall x$ *tel que* $x^{w_1} \in V \wedge x \in fv(\mathcal{A}_2)$*, et* $w_1 \notin L = Loc(\mathcal{A}_1)$*, alors toute occurence de* $x$ *est précédé d'un wait*$^{w_1}(X)\cdot$*, avec* $x \in X$.
– *Si* $\phi_{B_1}(e \rightarrow_\mu B_2, \mathbf{l}_1) = \left((\phi_{B_1}(e)) \rightarrow_\rho^\alpha R\right)$*, alors pour tout* $B_3$*, il existe R', tel que* $\phi_{B_1}(e \rightarrow_\mu B_3, \mathbf{l}_1) = \left((\phi_{B_1}(e)) \rightarrow_\rho^\alpha R'\right)$
– *Si hide* $(e) =$ *hide* $(f)$*, alors hide* $\left(\phi_{fst(e)}(e)\right) =$ *hide* $\left(\phi_{fst(f)}(f)\right)$.

*Démonstration.* On va maintenant définir $\phi_{B_1}$ de la manière suivante, par induction sur la longueur de l'exécution :
– si $e = \{B_1 \mid \mathcal{A}_1, \mathbf{l}_1\}$, alors $\phi_{B_1}(e) = B_1, \mathbf{l}_1$.
– si $e = f \rightarrow_\mu B_2, \mathbf{l}_2$. Par hypothèse d'induction, $\phi_{B_1}(f)$ est définie et vérifie les conditions désirées. Alors en particulier, si on note last $(\phi_{B_1}(f)) = B_3, \Phi, V, \mathbf{l}_1$, et last $(f) = B_4, \mathbf{l}_1$ on a que $B_4 \equiv \nu\overline{n} \cdot (B_3 \mid \mathcal{A}_2)$, avec $\mathcal{A}_2$ un processus modéliant un adversaire, c'est-à-dire congru à un processus sans frames, sans restrictions, sans contraintes, sans échange rapide.
  – si il existe une distribution $\gamma$, telle que $\mu = \nu\overline{n} \cdot (\gamma \mid \mathcal{A}_2), \mathbf{l}_2$, et $B_3, \mathbf{l}_1 \rightarrow \gamma, \mathbf{l}_2$, et $\mathbf{l}_1 = \mathbf{l}_2$ ou $\mathbf{l}_1 = \top$ (c'est-à-dire si il s'agit d'une réduction interne au processus (qui n'est pas une fin d'échange rapide) : l'adversaire n'intervient pas) : Alors il existe $B_5$, tel que $B_2 = \nu\overline{n} \cdot (B_5 \mid \mathcal{A}_2)$ et $\gamma(B_5) = \mu(B_2, \mathbf{l}_2)$, et on définit $\phi(e)$ de la manière suivante :
$$\phi(e) = \phi(f) \rightarrow^\tau_{(\gamma, \Phi, V, \mathbf{l}_2)} (B_5, \Phi, V, \mathbf{l}_2)$$
  – si $\mu = \delta_{\nu\overline{n}\cdot(B_5 \mid \mathcal{A}_3)}, \mathbf{l}_2$, et $\mathbf{l}_2 = \top$, et $\mathbf{l}_1 = \text{lock}\,(w_2)$, et $\mathcal{A}_2 \equiv \text{wait}^{w_2}(X) \cdot \mathcal{A}_3$ et $B_3 \equiv \text{end} \cdot B_5$ (c'est-à-dire qu'il s'agit d'une fin d'échange rapide) :
$$\phi(e) = \phi(f) \rightarrow^\tau_{\delta_{B_5, \Phi, \emptyset, \top}} (B_5, \Phi, \emptyset, \top)$$
  – si $\exists\gamma$, telle que $\mu = \nu\overline{n} \cdot (A_3 \mid \mathcal{C} \mid \gamma), \mathbf{l}_2$, avec $B_3 = A_3 \mid \mathcal{C}$ et où $A_3$ est un processus sans contraintes, et que de plus $\mathcal{A}_2 \mid \mathcal{C}, \mathbf{l}_2 \rightarrow \gamma \mid \mathcal{C}, \mathbf{l}_2$ (c'est-à-dire si il s'agit d'une réduction interne à l'adversaire : le processus n'intervient pas) :
    On remarque d'abord que la localisation n'est pas modifiée (en effet, l'adversaire ne dispose pas des primitives permettant de déclencher ou de finir un échange rapide.) Il y a deux possibilités :
    – ou l'adversaire a accompli une action non probabiliste, et $\gamma = \delta_{\mathcal{A}_3}$. On définit $\phi_{B_1}(e)$ de la manière suivante :
$$\phi_{B_1}(e) = \phi(f) \rightarrow^\tau_{\delta_{\text{last}\left(\phi_{B_1}(f)\right)}} (\text{last}\,(\phi_{B_1}(f)))$$
    – ou l'adversaire a accompli une action probabiliste, et c'est donc un test d'égalité. Alors il existe $M, N$ ground terms, tel que :$\gamma = p \cdot \delta_{((\mathcal{A}_3 \mid (M=N)))} + (1-p) \cdot \delta_{((\mathcal{A}_4 \mid \text{not}(M=N)))}$ :
    Donc $B_2 = ((\mathcal{A}_3 \mid (M = N)) \mid B_3)$, ou bien $B_2 = ((\mathcal{A}_4 \mid \text{not}\,((M = N)) \mid B_3))$. Si on est dans le premier cas :
$$\phi_{B_1}(e) = \phi(f) \rightarrow^{(\text{test}(M,N))}_{p\cdot\delta_{(B_3 \mid (M=N), \Phi\cup\text{Eq}(M=N), V, \mathbf{l}_2)} + (1-p)\cdot(\delta_{B_3 \mid \text{not}(M=N), \Phi\cup\text{NotEq}(M=N), V, \mathbf{l}_2)}} (B_3 \mid (M = N), \Phi\cup\text{Eq}(M = N), V, \mathbf{l}$$
    De même, si on est dans le deuxième cas :
$$\phi_{B_1}(e) = \phi(f) \rightarrow^{(\text{test}(M,N))}_{p\cdot\delta_{(B_3 \mid (M=N), \Phi\cup\text{Eq}(M=N), V, \mathbf{l}_2)} + (1-p)\cdot(\delta_{B_3 \mid \text{not}(M=N), \Phi\cup\text{NotEq}(M=N), V, \mathbf{l}_2)}} (B_3 \mid \text{not}\,(M = N), \Phi\cup\text{NotEq}(M =$$
  – si $B_3 \equiv ((\overline{p}(x) \cdot B_4 \mid B_5))$, $\mathcal{A}_2 \equiv (p(x) \cdot \mathcal{A}_3 \mid \mathcal{A}_4)$ et $\mu = \delta_{\nu\overline{n}\cdot B_4 \mid B_5 \mid \mathcal{A}_3 \mid \mathcal{A}_4, \mathbf{l}_2}$, et $\mathbf{l}_2 = \top$. Alors on définit :
$$\phi_{B_1}(e) = \phi(f) \rightarrow^{\overline{p}(x)}_{\left(\delta_{B_4 \mid B_5, V, \Phi, \top}\right)} B_4 \mid B_5, V, \Phi, \top$$
  – si $B_3 \equiv \nu x \cdot ((\overline{p}(x) \cdot B_4 \mid B_5))$, $\mathcal{A}_2 \equiv (p(x) \cdot \mathcal{A}_3 \mid \mathcal{A}_4)$ et $\mu = \delta_{\nu\overline{n}\cdot\nu x \cdot B_4 \mid B_5 \mid \mathcal{A}_3 \mid \mathcal{A}_4, \mathbf{l}_2}$, et $\mathbf{l}_2 = \top$. Alors on définit :
$$\phi_{B_1}(e) = \phi(f) \rightarrow^{\nu x \cdot \overline{p}(x)}_{\left(\delta_{B_4 \mid B_5, V, \Phi, \top}\right)} B_4 \mid B_5, V, \Phi, \top$$

– si $B_3 \equiv \left(\left(\overline{\text{fast}\,(\cdot)}\,(x)\cdot B_4 \mid B_5\right)\right)$, $\mathcal{A}_2 \equiv (\text{fast}\,(w_3)\,(x)\cdot \mathcal{A}_3 \mid \mathcal{A}_4)$ et $\mu = \delta_{\nu\overline{n}\cdot B_4|B_5|\mathcal{A}_3|\mathcal{A}_4,\mathbf{l}_2}$ si $w_3 = w_2$, et $\mu = \delta_{\nu\overline{n}\cdot B_4|B_5|\text{wait}^{w_2}\cdot\mathcal{A}_3|\mathcal{A}_4,\mathbf{l}_2}$ sinon et $\mathbf{l}_2 = \text{lock}\,(w_2)$. Alors on définit :

$$\phi_{B_1}(e) = \phi(f) \rightarrow^{\overline{\text{fast}(\cdot)}(x)}_{\left(\delta_{B_4|B_5,V,\Phi,\mathbf{l}_2}\right)} B_4 \mid B_5, V, \Phi, \mathbf{l}_2$$

– si $B_3 \equiv \nu x\cdot\left(\left(\overline{\text{fast}\,(\cdot)}\,(x)\cdot B_4 \mid B_5\right)\right)$, $\mathcal{A}_2 \equiv (\text{fast}\,(w_3)\,(x)\cdot \mathcal{A}_3 \mid \mathcal{A}_4)$ et $\mu = \delta_{\nu\overline{n}\cdot\nu x\cdot B_4|B_5|\mathcal{A}_3|\mathcal{A}_4,\mathbf{l}_2}$ si $w_3 = w_2$, et $\mu = \delta_{\nu\overline{n}\cdot\nu x\cdot B_4|B_5|\text{wait}^{w_2}\cdot\mathcal{A}_3|\mathcal{A}_4,\mathbf{l}_2}$ sinon et $\mathbf{l}_2 = \text{lock}\,(w_2)$. Alors on définit :

$$\phi_{B_1}(e) = \phi(f) \rightarrow^{\overline{\text{fast}(\cdot)}(x)}_{\left(\delta_{B_4|B_5,V\cup\{x\}^{w_2},\Phi,\mathbf{l}_2}\right)} B_4 \mid B_5, V\cup\{x\}^{w_2}, \Phi, \mathbf{l}_2$$

– Si $B_3 \equiv \nu\bar{t}\cdot((p\,(x)\cdot B_4 \mid B_5))$, et $\mathcal{A}_2 \equiv (\overline{p}\,(M)\cdot\mathcal{A}_3 \mid \mathcal{A}_4)$ (où $\mathcal{A}_3, \mathcal{A}_4$ sont des processus modélisant un adversaire.). et $\mu = \delta_{\nu\overline{n}\cdot\left(\nu\bar{t}\cdot B_4|B_5\right)|\mathcal{A}_3|\mathcal{A}_4|\{\{x/M\}\},\mathbf{l}_2}$. Alore on définit :

$$\phi_{B_1}(e) = \phi(f) \rightarrow^{p(M)}_{\left(\delta_{\nu\bar{t}\cdot(B_4\{M/x\})|B_5,V,\Phi,\mathbf{l}_2}\right)} (B_4\{M/x\}) \mid B_5, V, \Phi, \mathbf{l}_2$$

– Si $B_3 \equiv \nu\bar{t}\cdot((\text{fast}\,(w_2)\,(x)\cdot B_4 \mid B_5))$, et $\mathcal{A}_2 \equiv \left(\overline{\text{fast}\,(\cdot)}\,(M)\cdot\mathcal{A}_3 \mid \mathcal{A}_4\right)$ (où $\mathcal{A}_3, \mathcal{A}_4$ sont des processus modélisant un adversaire.)., $\mathbf{l}_2 = \text{lock}\,(w_2)$ et $\mu = \delta_{\nu\overline{n}\cdot\left(\nu\bar{t}\cdot B_4|B_5\right)|\mathcal{A}_3|\mathcal{A}_4|\{\{x/M\}\},\text{lock}(w_2)}$. Alore on définit :

$$\phi_{B_1}(e) = \phi(f) \rightarrow^{\text{fast}(w_2)(M),Loc\mathcal{A}_1}_{\left(\delta_{\nu\bar{t}\cdot(B_4\{M/x\})|B_5,V,\Phi,\mathbf{l}_2}\right)} (B_4\{M/x\}) \mid B_5, V, \Phi, \mathbf{l}_2$$

– Si $B_3 \equiv \nu\bar{t}\cdot((\text{fast}\,(w_3)\,(x)\cdot B_4 \mid B_5))$, et $\mathcal{A}_2 \equiv \left(\overline{\text{fast}\,(\cdot)}\,(M)\cdot\mathcal{A}_3 \mid \mathcal{A}_4\right)$ (où $\mathcal{A}_3, \mathcal{A}_4$ sont des processus modélisant un adversaire.), et $\mathbf{l}_2 = \text{lock}\,(w_2)$ et $w_2 \neq w_3$ et $\mu = \delta_{\nu\overline{n}\cdot\left(\nu\bar{t}\cdot\text{wait}^{w_2}\cdot B_4|B_5\right)|\mathcal{A}_3|\mathcal{A}_4|\{\{x/M\}\},\text{lock}(w_2)}$. Alore on définit :

$$\phi_{B_1}(e) = \phi(f) \rightarrow^{\text{fast}(w_2)(M),Loc(\mathcal{A}_1)}_{\left(\delta_{\nu\bar{t}\cdot(B_4\{M/x\})|B_5,V,\Phi,\mathbf{l}_2}\right)} \text{wait}^{w_2}\cdot(B_4\{M/x\}) \mid B_5, V, \Phi, \mathbf{l}_2$$

□

Soit $\sigma$ un scheduler sur les exécution non labellées. Alors on construit $\mathcal{S}$ de la manière suivante :Soit $e_l$ une exécution labellée.

– Si $e_l \in \bigsqcup_{B_1 \text{extended process avec contraintes}} \text{Im}(\phi_{B_1})$. (On remarque qu'il s'agit d'une union disjointe.) Alors soit $B_1, f$ respectivement l'unnique processus et exécution non labellée tels que $e_l = \phi_{B_1}(e)$. Soit $\mu = \sigma(e)$, et $s \in \text{Support}(\mu)$. Alors (par définiton de $\phi$), il existe $\rho$ une distribution, $\alpha$ une action, $t$ un état du lts, tels que $\phi_{B_1}(e \rightarrow_\mu s) = e_l \rightarrow^\alpha_\rho t$ (et de plus, $\alpha$ et $\rho$ sont les mêmes pour tout $s \in \text{Support}(\mu)$). On définit :

$$\mathcal{S}(e_l) = (\alpha, \rho)$$

– sinon,

$$\mathcal{S}(e_l) = (\alpha, \rho) = \perp$$

**Lemme 12.** *Pour tout ensemble $X \in \sigma Field^{\sigma}_{B_1|\mathcal{A}_1,\top}$,*

$$Prob^{\sigma}_{B_1|\mathcal{A}_1}(X) = LabProb^{\mathcal{S}}_{B_1}(\Phi_{B_1}(X))$$

*Démonstration.* $\Phi_{B_1}$ est une fonction injective, donc

$$g^{\sigma}_{B_1|\mathcal{A}_1,\top} : Y \in \sigma\text{Field}^{\sigma}_{B_1|\mathcal{A}_1,\top} \rightarrow P^{\mathcal{S}}_{B_1}(\Phi_{B_1}(Y))$$

est une mesure sur $\sigma\text{Field}^{\sigma}_{B_1|\mathcal{A}_1,\top}$.
De plus, les conditions imposées sur $Phi_{B_1}$ impliquent que : pour toute exécution non labéllée $e \in Exec^{\sigma}_{B_1|\mathcal{A}_1}$,

$$P^{\sigma}_{B_1|\mathcal{A}_1,\top}((\uparrow e)) = P^{\mathcal{S}}_{B_1}(\Phi_{B_1}(\uparrow(e)))$$

puisque $P^{\sigma}_{B_1|\mathcal{A}_1,\top}$ et $g^{\sigma}_{B_1|\mathcal{A}_1,\top}$ sont deux mesures sur $\sigma\text{Field}^{\sigma}_{B_1|\mathcal{A}_1,\top}$ qui coincident sur $(\uparrow e)$ pour toute exécutions $e$, ces deux fonctions sont égales. □

On va maintenant prouver le lemme : Soit $B_1 \in Process^{\textbf{result}}$

$$\begin{aligned}
Pr(\mathcal{A}_1 \mid B_1, \mathbf{l}_1 \rightarrow_\sigma^{\leq n} F_r) &= Prob^\sigma_{\mathcal{A}_1|B_1}\{e \in Exec^\sigma_{\mathcal{A}_1|B_1} \mid \exists i \leq n, e^i \in F_r\} \\
&= lProb^\sigma_{B_1}\left(\Phi_{B_1}(\{e \in Exec^\sigma_{\mathcal{A}_1|B_1} \mid \exists i \leq n, e^i \in F_r\})\right) \\
&\leq lProb^\sigma_{B_1}\left(\{e_l \in lExec^{\mathcal{S}}_{B_1} \mid \exists i \leq n, {e_l}^i = (B_2, V, \Phi, \mathbf{l}_1) \text{ tel que } \exists \mathcal{A}_2 \in Adv_L^{\textbf{result}},\ (B_2 \mid \mathcal{A}_2) \in F_r\}\right) \\
&\leq Pr(B_1 \rightarrow_\sigma^{\leq n} E_r)
\end{aligned}$$

□

**Lemme 13** (Stratégie ⇒ Contexte)**.** *Il existe* $\Theta$ *telle que :*

$$\Theta : A_1 \in \{\textit{extended process sans contrainte }\}, V, \Phi, \mathbf{l}_1, n \in \mathbb{N}, \mathcal{S}\ \textit{stratégie} \rightarrow A \in Adv, \sigma \in \{\textit{Schedulers}\}$$

*, telle que :*

$\forall E, \forall s = (A_1, \Phi, V, \mathbf{l}_1)$*, si* $\Theta(s, n, \mathcal{S}) = Adv_1, \sigma$*, alors*

– $\forall \mathcal{C}$ *contrainte,*

$$\begin{aligned}
Pr(Adv_1 \mid A_1 \mid \mathcal{C}, \mathbf{l}_1 \rightarrow_\sigma^{\leq n} \{(B_3, \mathbf{l}_1) \mid \exists B_4, \mathcal{A}_3, \Psi, V_2 \textit{ tels que } B_3 \equiv B_4 \mid \mathcal{A}_3 \wedge (B_4, \Psi, V_2, \mathbf{l}_2) \in E\}) \\
\geq Pr(A_1 \mid \mathcal{C}, V, \Phi, \mathbf{l}_1 \rightarrow_{\mathcal{S}}^{\leq n} E)
\end{aligned}$$

– $fn(Adv_1) \subseteq \{m \in Names \mid \exists e, \alpha \textit{ tel que } \mathcal{S}(e) = (\alpha, \mu) \wedge m \in fn(\alpha)\}$

*Démonstration.* Le raisonnement se fait par récurrence sur $n \in \mathbb{N}$ : si $n = 0$, on prend comme adversaire le processus nul : $\mathcal{A}_1 = 0$. Sinon, on s'intéresse à l'état $t$ dans lequel on arrive après une étape d'exécution, et on construit par hypothèse d'induction un adversaire $\mathcal{A}_2$ qui simule la stratégie à partir de $t$. Ensuite, en fonction de quel règle de réduction a été appliqué, on construit $\mathcal{A}_1$ à partir de $\mathcal{A}_2$.

On introduit d'abord la notation suivante : Soit $\mathcal{S}$ une stratégie, soit $s \in Hidden$, $\mu \in$ Distributions $(Hidden)$, $e$ une exécution, on note $\text{Next}_{\mathcal{S}}((s, \alpha, \mu))$ la stratégie $\mathcal{T}$ définie par : si il existe $t, \mu'$ tel que hide $(t) = s$ et hide $(\mu') = \mu$, et $t \rightarrow_\mu^\alpha e$ est une exécution, alors $\mathcal{T}(e) = \mathcal{S}(t \rightarrow_{\mu'}^\alpha e)$, et $\mathcal{T}(f) = \perp$ sinon. On notera également $\mathcal{T} = \mathcal{S}(s \rightarrow_\mu^\alpha e)$ (par abus de notation, puisques $\rightarrow_\mu^\alpha e$ n'est pas une exécution. )

On impose en fait à $\Theta$ les conditions plus restrictives suivantes :

– Soit $s = A_1, \Phi, V, \mathbf{l}_1$, $(\mathcal{A}_1, \sigma) = \Theta(s, n, \mathcal{S})$. On définit $\mathcal{T}$ de la manière suivante : Si $\mathcal{S}(\{A_1, \Phi, V, \mathbf{l}_1\}) = (\alpha, \mu)$, $\mathcal{T}(e) = \mathcal{S}(s \rightarrow^\alpha_{\text{hide}(\mu)} e)$ et la condition suivante doit être vérifiée : $\forall B_1$ tel que hide $(B_1) = A_1$, si $\mathcal{S}(\{B_1, \Phi, V, \mathbf{l}_1\}) = (\alpha, \rho)$ et pour tout $B_2$ tel que $\rho(B_2, \Psi, V_2, \mathbf{l}_2) = p > 0$

$$\sigma(B_1 \mid \mathcal{A}_1, \mathbf{l}_1) = \gamma$$

et $\exists \bar{t},$ et Aux un processus sans frame ni contrainte , (tel que de plus $(\nu \vec{\cdot} t\,(B_2 \mid \mathcal{A}_2 \mid \text{Aux}, \mathbf{l}_2))$ correspond à un unique $(B_2, \Psi, V_2, \mathbf{l}_2) \in$ Support $(\mu)$)

$$\gamma(\nu \vec{\cdot} t\,(B_2 \mid \mathcal{A}_2 \mid \text{Aux}, \mathbf{l}_2)) = p$$

, et de plus $\sigma(B_1, \mathbf{l}_1 \rightarrow (\nu \vec{\cdot} t\,(B_2 \mid \mathcal{A}_2 \mid \text{Aux}, \mathbf{l}_2) \rightarrow \nu \bar{t} \cdot e \mid \text{Aux})) = \nu \bar{t} \cdot \rho(B_2 \mid \mathcal{A}_2, \mathbf{l}_2 \rightarrow e) \mid \text{Aux}$

– Pour tout $\{x\}^{w_1} \in V$, et $w_1 \notin Loc(\mathcal{S})$ toute occurence libre de $x$ dans $\mathcal{A}_1$ est précédé de wait$^{w_1}(x)\cdot$.

On va construire $\Theta$ par induction sur $n$ :

$n = 0$ : $\Theta(\cdot, 0, \cdot) = (0, \sigma : (\cdot) \rightarrow \perp)$

$n \to n+1$ : Soit $A_1$ un processus sans contraintes, $\Phi$ représentant les contraintes connues par l'adversaire, $V$ représentant les variables dont l'utilisation est restreinte, $\mathbf{l}_1 \in$ Location constraints. On va définir $\Theta((A_1, \Phi, V, \mathbf{l}_1), \mathcal{S}, n)$. On sait que le résultat d'une stratégie ne dépend pas des contraintes. Donc si on note, pour tout processus $B_3$ tel que hide $(B_3) = A_1$, $(\alpha_{B_3}, \mu_{B_3}) = \mathcal{S}(\{B_3, \Phi, V, \mathbf{l}_1\})$, on peut définir $\alpha_{B_3} = \alpha$, et hide $(\mu_{B_3}) = \mu$. On remarque de plus qu'on doit avoir (d'après les règles de réduction), que $\mu = (\mu', \mathbf{l}_2)$ (c'est-à-dire que la localisation à l'étape suivante n'est pas probabiliste).
On va définir une stratégie $\mathcal{T}$, qui correspond à la stratégie $\mathcal{S}$, lorsqu'on a déjà effectué la première étape. On définit la stratégie $\mathcal{T}$ de la manière suivante : $\forall B_3$ tel que hide $(B_3) = A_1, \forall s \in$ Support $(\mu_{B_3})$, pour toute exécution e (qui représente la suite de l'exécution)

$$\mathcal{T}(s \to e) = \mathcal{S}\left(A_3, \Phi, V, \mathbf{l}_1 \to_\mu^\alpha s \to e\right)$$
$$\mathcal{T}(f) = \bot \text{ si hide } (first(f)) \notin \text{Support } (\mu)$$

- Si $\alpha = \tau$. Il y a trois possibilités :
  - Si $\mathbf{l}_1 = \mathbf{l}_2$ :
    Alors Support $(\mu)$ est fini, et pour $s_i = \left(A_4^i, \Psi, V_2, \mathbf{l}_2\right) \in$ Support $(\mu)$, par hypothèse d'induction, $\Theta((s_i), n, \mathcal{T})$ est déjà défini. Soit $(Adv_i, \sigma_i) = \Theta((s_i), n, \mathcal{T})$. On pose

    $$Adv = ||_{s_i \in \text{Support}(\mu)} Adv_i$$

    et on définit $\sigma$ de la manière suivante :
    - $\forall B_3$ tel que hide $(B_3) = A_3, \forall (B_4^i)$ tel que $\left(B_4^i, \Psi, V_2, \mathbf{l}_2\right) \in$ Support $(\mu_{B_3})$, on note $k_i$ tel que hide $\left(B_4^i\right) = A_4^{k_i}$, et on définit :

      $$\sigma(\{B_3 \mid Adv, \mathbf{l}_1\}) = \sum_{1 \le i \le n} \mu_{B_3}(B_4^i, \Psi, V_2, \mathbf{l}_2)\delta_{B_4^i \mid Adv, \mathbf{l}_2}$$
      $$\sigma((\{B_3 \mid Adv, \mathbf{l}_1\}) \to_{\sum_{1 \le i \le n} \mu_{B_3}(B_4^i, \Psi, V_2, \mathbf{l}_2)\delta_{B_4^i \mid Adv, \mathbf{l}_2}} B_4^i \mid Adv, \mathbf{l}_2 \to e \mid (||_{j \neq i} Adv_j))$$
      $$= \left(\sigma_{k_i}(B_4^i \mid Adv_i, \mathbf{l}_2 \to e)\right) \mid (||_{j \neq i} Adv_j)$$

      où, pour $P$ un processus sans contraintes et sans substitutions, $e \mid P$ représente une exécution de First $(e \mid P)$, dans laquelle $P$ n'intervient jamais.
    - $\sigma(f) = \bot$ sinon.
  - Si $\mathbf{l}_1 = \top, \mathbf{l}_2 = \text{lock}(w_1)$.
    Alors $V = \emptyset$, et il existe $\bar{t}, A_4, A_5$ tels que : $A_3 = \nu\bar{t} \cdot \text{begin}(w_1) \cdot A_4 \mid A_5$, et $\mu = \delta_{\nu\bar{t} \cdot A_4 \mid A_5, \Phi, \emptyset, \text{lock}(w_1)}$ et soit $(\mathcal{A}_2, \rho) = \Theta((A_4, \Phi, \emptyset, \mathbf{l}_2), n, \mathcal{T})$. Alors on définit :

    $$Adv = \mathcal{A}_2$$
    $$\sigma(\{\nu\bar{t} \cdot (\text{begin}(w_1) \cdot A_4 \mid A_5 \mid \mathcal{C} \mid Adv), \top\}) = \delta_{\nu\bar{t} \cdot (A_4 \mid A_5 \mid \mathcal{C} \mid Adv), \text{lock}(w_1)}$$
    $$\sigma(\nu\bar{t} \cdot (\text{begin}(w_1) \cdot A_4 \mid A_5 \mid \mathcal{C} \mid Adv) \to e) = \rho(e)$$

  - Si $\mathbf{l}_1 = \text{lock}(w_1), \mathbf{l}_2 = \top$
    Alors $A_3$ est de la forme : $A_3 \equiv \left(\nu\bar{t} \cdot \text{end} \cdot A_4 \mid A_5\right)$ et que de plus $\mu = \delta_{(A_4 \mid A_5, \Phi, \emptyset, \top)}$. Alors on définit $\mathcal{T}$ comme dans les cas précédents, et on note :

    $$(\mathcal{A}_2, \rho) = \Theta((A_4 \mid A_5, \Phi, \emptyset, \top), n, \mathcal{T})$$

    . Et on pose

    $$\text{Adv} = \text{wait}^{w_1}(V) \cdot \mathcal{A}_2$$
    $$\sigma(\{\nu\bar{t} \cdot (\text{end} \cdot A_4 \mid A_5 \mid \mathcal{C} \mid \text{Adv}), \text{lock}(w_1)\}) = \delta_{\nu\bar{t} \cdot (A_4 \mid A_5 \mid \mathcal{C} \mid \mathcal{A}_2), \top}$$
    $$\sigma(\nu\bar{t} \cdot (\text{end} \cdot w_1 A_4 \mid A_5 \mid \mathcal{C} \mid \text{Adv}, \text{lock}(w_1)) \to e) = \rho(e)$$

– Si $\alpha = \overline{p}(x)$. Alors $\mathbf{l}_1 = \top$ et de plus $A_3$ est de la forme : $A_3 \equiv \nu\bar{t} \cdot (\overline{p}\,(x) \cdot A_4 \mid A_5)$ avec $x \notin t$, et de plus $\mu = \delta_{\nu\bar{t}\cdot(A_4|A_5,\Phi,V,\top)}$. On peut appliquer l'hypothèse d'induction :

$$(\mathcal{A}_2, \rho) = \Theta(\left(\nu\bar{t} \cdot (A_4 \mid A_5)\right), \Phi, V, \top), n, \mathcal{T})$$

On pose

$$Adv = p\,(x) \cdot \mathcal{A}_2$$

et on définit $\sigma$ de la manière suivante :
– Pour tout $B_3$ tel que hide $(B_3) \equiv A_3$, c'est-à-dire : $B_3 \equiv \nu\bar{t} \cdot \overline{p}\,(x) \cdot A_4 \mid A_5 \mid \mathcal{C}$

$$\sigma(\{B_3 \mid Adv\}) = \delta_{\nu\bar{t}\cdot A_4|A_5|\mathcal{C}|\mathcal{A}_2}$$

et

$$\sigma(B_3 \mid Adv \to e) = \rho(e)$$

– $\sigma(e) = \bot$ si hide $(first(e)) \neq A_3 \mid Adv$

– Si $\alpha = \nu x \cdot \overline{p}(x)$, et que $A_3$ est de la forme : $A_3 \equiv \nu\bar{t} \cdot \nu x \cdot (\overline{p}\,(x) \cdot A_4 \mid A_5)$ avec $x \notin t$, et que de plus $\mu = \delta_{\nu\bar{t}\cdot(A_4|A_5,\Phi,V,\mathbf{l}_1)}$, et $\mathbf{l}_1 = \top$.
On peut maintenant appliquer l'hypothèse d'induction :

$$(\mathcal{A}_2, \rho) = \Theta(\left(\nu\bar{t} \cdot (A_4 \mid A_5)\right), \Phi, V, \top), n, \mathcal{T})$$

On pose

$$Adv = p\,(x) \cdot \mathcal{A}_2$$

et on définit $\sigma$ de la manière suivante :
– Pour tout $B_3$ tel que hide $(B_3) \equiv A_3$, c'est-à-dire : $B_3 \equiv \nu x \cdot \nu\bar{t} \cdot \overline{p}\,(x) \cdot A_4 \mid A_5 \mid \mathcal{C}$

$$\sigma(\{B_3 \mid Adv\}) = \delta_{\nu x\cdot\left(\nu\bar{t}\cdot A_4|A_5|\mathcal{C}|\mathcal{A}_2\right)}$$

et

$$\sigma(B_3 \mid Adv \to \nu x \cdot e) = \nu x \cdot \rho(e)$$

– $\sigma(e) = \bot$ si hide $(first(e)) \neq A_3 \mid Adv$

– Si $\alpha = \overline{\text{fast}\,(\cdot)}(x)$, et que $A_3$ est de la forme : $A_3 \equiv \nu\bar{t} \cdot \left(\overline{\text{fast}\,(\cdot)}\,(x) \cdot A_4 \mid A_5\right)$ avec $x \notin t$, et que de plus $\mu = \delta_{\nu\bar{t}\cdot(A_4|A_5,\Phi,V\mathbf{l}_1)}$, et $\mathbf{l}_1 = \text{lock}\,(w_1)$. On peut maintenant appliquer l'hypothèse d'induction : $(\mathcal{A}_2, \rho) = \Theta(\left(\nu\bar{t} \cdot (A_4 \mid A_5)\right), \Phi, V, \text{lock}\,(w_1))\,, n, \mathcal{T})$. Il y a deux possibilités :
– $w_1 \in Loc\,(\mathcal{S})$ : On pose

$$Adv = \text{fast}\,(w_1)\,(x) \cdot \mathcal{A}_2$$

et on définit $\sigma$ de la manière suivante :
– Pour tout $B_3$ tel que hide $(B_3) \equiv A_3$, c'est-à-dire : $B_3 \equiv \nu\bar{t} \cdot \overline{\text{fast}\,(w_1)}\,(x) \cdot A_4 \mid A_5 \mid \mathcal{C}$

$$\sigma(\{B_3 \mid Adv\}) = \delta_{\nu\bar{t}\cdot A_4|A_5|\mathcal{C}|\mathcal{A}_2}$$

et

$$\sigma(B_3 \mid Adv \to e) = \rho(e)$$

– $\sigma(e) = \bot$ si hide $(first(e)) \neq A_3$

– $w_1 \notin Loc\mathcal{S}$. On remarque que l'on a par construction $\text{wait}^{w_1}\,(y) \cdot \mathcal{A}_2 \equiv \mathcal{A}_2$ (parce que $y \notin fv(\mathcal{A}_2)$). On peut par conséquent définir : (avec $y \notin fv(\mathcal{A}_2)$) (l'idée générale est que $\mathcal{A}_2$ n'a pas besoin de se servir du terme qu'il va recevoir, puisque $x$ n'est pas restreint : il peut utiliser directement $x$)

$$Adv = \text{fast}\,(\cdot)\,(y) \cdot \mathcal{A}_2$$

. et on définit $\sigma$ de la manière suivante :

– Pour tout $B_3$ tel que hide $(B_3) \equiv A_3$, c'est-à-dire : $B_3 \equiv \nu\bar{t} \cdot \overline{\text{fast}\,(w_1)}\,(x) \cdot A_4 \mid A_5 \mid \mathcal{C}$

$$\sigma(\{B_3 \mid Adv\}) = \delta_{\nu\bar{t}\cdot(A_4|A_5|\mathcal{C}|\mathcal{A}_2)}$$

et

$$\sigma(B_3 \mid Adv \to e) = \rho(e)$$

– $\sigma(e) = \bot$ si hide $(first(e)) \neq A_3$

– Si $\alpha = \nu x \cdot \overline{\text{fast}\,(\cdot)}(x)$, et que $A_3$ est de la forme : $A_3 \equiv \nu x \cdot \nu\bar{t} \cdot \left(\overline{\text{fast}\,(\cdot)}\,(x) \cdot A_4 \mid A_5\right)$ avec $x \notin t$, et que de plus $\mu = \delta_{\nu\bar{t}\cdot\left(A_4|A_5,\Phi,V\cup\{x\}_1^w,\mathbf{l}_1\right)}$, et $\mathbf{l}_1 = \text{lock}\,(\mathbf{l}_1)$. On peut maintenant appliquer l'hypothèse d'induction : $(\mathcal{A}_2, \rho) = \Theta(\left(\nu\bar{t} \cdot (A_4 \mid A_5)\right., \Phi, V \cup \{x\}_1^w, \text{lock}\,(w_1))\left.\right), n, \mathcal{T})$. Il y a deux possibilités :

– $w_1 \in Loc\,(\mathcal{S})$ : On pose

$$Adv = \text{fast}\,(w_1)\,(x) \cdot \mathcal{A}_2$$

et on définit $\sigma$ de la manière suivante :

– Pour tout $B_3$ tel que hide $(B_3) \equiv A_3$, c'est-à-dire : $B_3 \equiv \nu x \cdot \nu\bar{t} \cdot \overline{\text{fast}\,(w_1)}\,(x) \cdot A_4 \mid A_5 \mid \mathcal{C}$

$$\sigma(\{B_3 \mid Adv\}) = \delta_{\nu\bar{t}\cdot A_4|A_5|\mathcal{C}|\mathcal{A}_2}$$

et

$$\sigma(B_3 \mid Adv \to \nu x \cdot e) = \nu x \cdot \rho(e)$$

– $\sigma(e) = \bot$ si hide $(first(e)) \neq A_3$

– $w_1 \notin Loc\mathcal{S}$. On remarque que l'on a par hypothèse d'induction $\text{wait}^{w_1}\,(x) \cdot \mathcal{A}_2 \equiv \mathcal{A}_2$ (parce que $x$ est une variable restreinte pour les locations que l'adversaire possède, donc toute occurence de $x \in \mathcal{A}_2$ est déjà précédé d'un $\text{wait}^{w_1}\,(x)\cdot$). On peut par conséquent définir :

$$Adv = \text{fast}\,(\cdot)\,(x) \cdot \mathcal{A}_2$$

. et on définit $\sigma$ de la manière suivante :

– Pour tout $B_3$ tel que hide $(B_3) \equiv A_3$, c'est-à-dire : $B_3 \equiv \nu\bar{t} \cdot \left(\overline{\text{fast}\,(w_1)}\,(x) \cdot A_4 \mid A_5 \mid \mathcal{C}\right)$

$$\sigma(\{B_3 \mid Adv\}) = \delta_{\nu\bar{t}\cdot(A_4|A_5|\mathcal{C}|\mathcal{A}_2)}$$

et

$$\sigma(B_3 \mid Adv \to e) = \nu x \cdot \rho(e)$$

– $\sigma(e) = \bot$ si hide $(first(e)) \neq A_3$

– Si $\alpha = \text{test}(M, N)$,alors $\mu = p \times \delta_{(A_3,\Phi\wedge(M=N),V,\mathbf{l}_1)} + (1-p) \times \delta_{(A_3,\Phi\wedge\text{not}((M=N)),V,\mathbf{l}_1)}$. On note :

$$(\mathcal{A}_2^1, \rho^1) = \Theta((A_3, \Phi \wedge (M = N), V, \mathbf{l}_1), n, \mathcal{T})$$

et

$$(\mathcal{A}_2^2, \rho^2) = \Theta((A_3, \Phi \wedge \text{not}\,((M = N)), V, \mathbf{l}_1), n, \mathcal{T})$$

Et on pose :

$$\mathcal{A}_1 = \text{ if } (M = N) \text{ then } \mathcal{A}_2^1 \text{ else } \mathcal{A}_2^2$$

et on définit $\sigma$ de la manière suivante :

– Pour tout $B_3$ tel que hide $(B_3) \equiv A_3$,

$$\sigma(\{B_3 \mid Adv, \mathbf{l}_1\}) = p \times \delta_{B_3|(M=N)|\mathcal{A}_2^1,\mathbf{l}_1} + (1-p) \times \delta_{B_3|\text{not}(M=N)|\mathcal{A}_2^2,\mathbf{l}_1}$$

et

$$\sigma(B_3 \mid Adv, \mathbf{l}_1 \to_{\sigma(\{B_3|Adv,\mathbf{l}_1\})} B_3 \mid \mathcal{A}_2^1, \mathbf{l}_1 \to e) = \rho^1(e)$$
$$\sigma(B_3 \mid Adv, \mathbf{l}_1 \to_{\sigma(\{B_3|Adv,\mathbf{l}_1\})} B_3 \mid \mathcal{A}_2^2, \mathbf{l}_1 \to e) = \rho^2(e)$$

– $\sigma(e) = \bot$ si hide $(first(e)) \neq A_3$

– Si $\alpha = \text{fast}\,(w_2)(x), S$, et que $\mathbf{l}_1 = \text{lock}\,(w_1)$. Alors $A_3$ est de la forme : $A_3 \equiv \nu\bar{t} \cdot (\text{fast}\,(w_2)\,(x) \cdot A_4 \mid A_5)$ avec $x \notin t$, et que de plus $\mu = \delta_{\nu\bar{t}\cdot(A_4\{x/M\}|A_5,\Phi,V\mathbf{l}_1)}$. On peut maintenant appliquer l'hypothèse d'induction : $(\mathcal{A}_2, \rho) = \Theta(\left(\nu\bar{t} \cdot (A_4\{x/M\} \mid A_5)\,, \Phi, V, \text{lock}\,(w_1)\right), n, \mathcal{T})$.
On pose

$$Adv = \overline{\text{fast}\,(\cdot)}\,(M) \cdot \mathcal{A}_2$$

et on définit $\sigma$ de la manière suivante :

– Pour tout $B_3$ tel que hide $(B_3) \equiv A_3$, c'est-à-dire : $B_3 \equiv \nu\bar{t} \cdot \overline{\text{fast}\,(w_1)}\,(x) \cdot A_4 \mid A_5 \mid \mathcal{C}$

– on définit le scheduler de la manière suivante :

– Si $w_1 = w_2$ :

$$\sigma(\{B_3 \mid Adv\}) = \delta_{\nu\bar{t}\cdot A_4\{x/M\}|A_5|\mathcal{C}|\mathcal{A}_2}$$

et

$$\sigma(B_3 \mid Adv \to e) = \rho(e)$$

$$\sigma(e) = \bot \text{ si hide } (first(e)) \neq A_3$$

– Si $w_1 \neq w_2$ :

$$\sigma(\{B_3 \mid Adv\}) = \delta_{\nu\bar{t}\cdot \text{wait}^{w_1}(x)\cdot(A_4\{x/M\}|A_5|\mathcal{C})|\mathcal{A}_2}$$

et

$$\sigma(B_3 \mid Adv \to e) = \rho(e)$$

$$\sigma(e) = \bot \text{ si hide } (first(e)) \neq A_3$$

□

# I Borne sur une attaque à distance pour une version modifiée du protocole Hancke-Kuhn

On s'intéresse à un langage dont la signature contient l'ensemble $\mathcal{L}$ de constructeurs suivant :

– $\mathbf{un}, \mathbf{zero}, \mathbf{concat}$, qui correpondent aux opérations élémentaires sur les chaînes de caractère

– un constructur $\mathbf{hash}$, qui correpond à une fonction de hashage, d'arité 1 et de fonction de longueur : $\left(p \mapsto \frac{2\cdot p}{3}\right)$

– un constructeur $F$, d'arité 2, qui correpond à la fonction calculée de la manière suivante : le $i$-eme bit de $F(k, a, b)$ est calculé de la manière suivante :

$$(F(k,a,b))_i = \begin{cases} a_i \text{ si } k_i = 0 \\ b_i \text{ si } k_i = 1 \end{cases} \quad \text{avec } a_i,\ b_i,\ k_i \text{ respectivement les } i\text{-eme bit de } a,\ b,\ k$$

On s'intéresse à une interprétation de la signature de ce langage tel que :

$$\mathscr{D}_{sig} = \mathscr{D}_{\sigma\setminus\mathcal{L}} \times \mathscr{D}^{\text{unif}}_{\mathbf{hash}} \times \mathscr{D}^{\text{nat}}_{\mathbf{un},\mathbf{zero},\mathbf{concat},F}$$

, où la distribution naturelle sur $\mathbf{un}, \mathbf{zero}, \mathbf{concat}, F$ correspond à la distribution où l'implémentation naturelle de ces constructeurs a une probabilité 1 (on la définit de manière analogue à la distribution naturelle pour le langage **Strings**). On choisit $\mathscr{D}_{names}$ comme la distribution uniforme pour les names, et $\mathscr{M} = \mathscr{M}(\mathscr{D}_{sig}, \mathscr{D}_{names})$

## I.1 Protocole

On considère le protocole ci-dessous :
Processue modélisant le Reader :

$$
\begin{aligned}
R_{s,w_1}^{\textbf{result}} = \\
&\nu N_A^{(p\mapsto p)} \cdot \nu N_B^{(p\mapsto p)} \\
&\cdot p\left(y^{(p\mapsto p)}\right) \cdot \\
&\overline{p}\left(N_A^{(p\mapsto p)}\right) \cdot \\
&\text{begin}\,(w_1) \cdot \overline{\text{fast}(p,w_1)}\left(N_B^{(p\mapsto p)}\right) \cdot \text{fast}(p,w_1)\left(z^{(p\mapsto p)}\right) \cdot \text{end} \cdot \\
&\text{if } \left(z^{(p\mapsto p)} = F(N_C^{(p\mapsto p)}, \mathbf{hash}(\mathbf{concat}(s, \mathbf{concat}(N_A^{(p\mapsto p)}), (y^{(p\mapsto p)}))))\right) \text{ then} \\
&\overline{\textbf{result}}\,(\mathbf{un}) \cdot 0 \text{ else} \\
&\overline{\textbf{result}}\,(\mathbf{zero}) \cdot 0
\end{aligned}
$$

Processus modélisant le Tag :

$$
\begin{aligned}
T_{s,w_1} = \\
&\nu N_B^{(p\mapsto p)} \cdot \overline{p}\left(N_B^{(p\mapsto p)}\right) \cdot p\left(x^{(p\mapsto p)}\right) \cdot \\
&\text{fast}(p,w_1)\left(y^{(p\mapsto p)}\right) \cdot \overline{\text{fast}(p,w_1)}\left(F(y^{(p\mapsto p)}, \mathbf{hash}(\mathbf{concat}(s, \mathbf{concat}(x^{(p\mapsto p)}, N_B^{(p\mapsto p)}))))\right) \cdot 0
\end{aligned}
$$

On a montré précédemment qu'il y a une attaque à distance qui s'éxécute en temps linéaire en $l$, et qui réussit avec probabilité $\left(\frac{3}{4}\right)^l$. On va montrer que :

**Théorème 4.** *Pour tout adversaire $\mathscr{A}$, il exise un polynome $P$, tel que : pour tout paramètre de sécurité $l$,*

$$Pr_l(\mathscr{A} \textit{ gagne pour l'attaque à distance }) \leq P(l) \cdot \left(\frac{3}{4}\right)^l$$

On va uttiliser la modélisation de l'adversaire par une stratégie dans le système de transition probabiliste :
Soit $\mathcal{S}$ une stratégie sur le lts labbellé, tel que $w_1 \notin Loc\,(\mathcal{S})$. On impose de plus la conditions suivante : pour chaque terme $M$ tel que $p(M)$, ou test$(M,N)$ est une action prescrite par la stratégie $\mathcal{S}_l$, alors la longueur syntaxique de $M$(qui est en particulier supérieur au nombre d'occurence de $h$ dans $M$) est inférieure à $P(l)$.

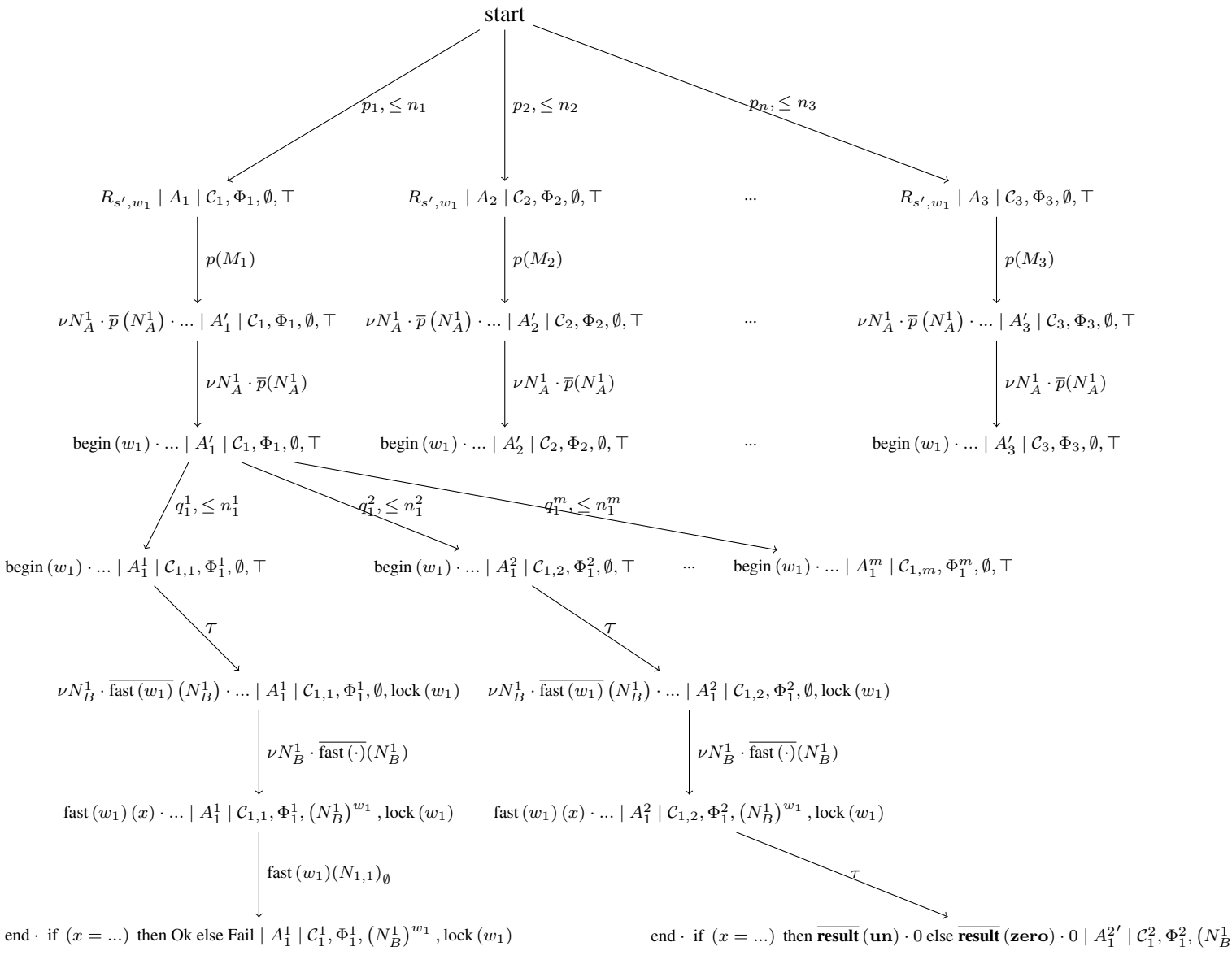
- On remarque d'abord que toute communication en dehors d'une phase rapide peut être simulée par la succession de deux actions labellées (et avec un label différent de $\tau$). Puisque on n'est intéressé par un nombre d'étapes d'éxécution borné par un polynome, on peut ne pas considérer les communications modéllisées par des $\tau$-actions en dehors des phases rapides. Ce n'est pas le cas pendant la phase rapide. (par exemple, un tag témoin (donc avec une clé différente), ayant la même localisation que le lecteur, peut intéragir avec lui, mais l'adversaire ne peut pas intervenir).

La modélisation de l'attaque à distance correspond à deux phases : une phase d'apprentissage où l'adversaire a accès à des tags et des lecteurs à la même localisation que lui, et une deuxième phase pendant laquelle il lance son attaque. On va ici donner encore plus de pouvoirs à l'adversaire : on suppose qu'il connaît tous les secrets de tous les protocoles : la phase 1 devient alors inutile, puisqu'il peut la simuler par lui-même. On va donc considérer l'état de départ suivant :

$$\text{start} = R_{s',w_1} \mid !\,(R_{s_2,w_1}\{p/\text{result'}\}) \mid !\,(T_{s_2,w_1})$$

On fixe $l$ un paramètre de sécurité : Pour toute stratégie, l'ensembles des traces d'éxécution possibles à partit de l'état $start$ peuvent être représentés par :

– l'étiquette $(p_i, \leq n_i)$ signifie qu'il s'agit en fait de $n_i$ transitions successives de probabilité dont le produit est égal à $p_i$, et qui peuvent faire intervenir les agents "témoins", mais pas le lecteur que l'adversaire a pris pour cible.
– les contraintes $\mathcal{C}_i$ ont la propriété suivante : $I_i = \{\eta \mid \eta(\mathcal{C}_i) \text{ est vrai } \}$ forment une pratition de l'ensemble des implémentations, et de plus $\sum_i p_i = 1$.
– les contraintes $\mathcal{C}_{i,j}$ ont la propriété suivante : Pour tout $i$, les ensembles $I_i, j = \{\eta \mid \eta(\mathcal{C}_{i,j}) \text{ est vrai } \}$ forment une partition de $I_i$




En effet, pendant la phase rapide, après que le lecteur ait émis son challenge, l'adversaire peut décider :
– de fournir un terme au lecteur (mais dans ce cas il ne peut pas uttiliser le challenge qui vient d'être émis, puisque l'adversaire n'a pas la localisation $w_1$). Dans ce cas, l'action effectuée en respectant la stratégie $\mathcal{S}$ est $\text{fast}(p, w_1)(N)_\emptyset$.
– de faire intéragir l'adversaire avec un tag honnête (mais avec une clé différente), qui se trouve au bon endroit). Dans ce cas, puisque le tag est honnête, il suit le protocol, et le terme recu par le lecteur comme réponse au challenge est forcément $F(N_B^{(p\mapsto p)}, \mathbf{hash}(\text{Concat}(s,')\text{Concat}(N, c^{(p\mapsto p)})))$ avec $c \in \mathcal{N}$, et $N$ un terme qui a été auparavant fourni par l'adversaire à ce tag, et $s' \neq s$. Dans ce cas, l'action effectuée en respectant la stratégie est l'action $\tau$.

La probabilité de succès de l'attaque est :

$$\sum_{i,j} p_i \times q_{i,j} \times \left[L_i = N_{i,j} \mid \mathcal{C}_i^j\right]_l$$

où on a uttilisé les notations suivantes :

- $L_i = F\left(N_B^{(p \mapsto p)}, \left(\mathbf{hash}(s, N_A^{(p \mapsto p)}, M_i)\right)\right)$
- – ou bien $N_B^{(p \mapsto p)} \notin N_{i,j}$ (si ce n'est pas une $\tau$-action) : on note $I_1$ l'ensemble des $(i,j)$ qui sont dans ce cas.
  - ou bien $N_{i,j} = \text{Choice}\left(N_B^1, split^1\left(h(s', P_{i,j}, c)\right), split^2\left(h(s', P_{i,j}, c)\right)\right)$ (et $N_B^1 \notin P_{i,j}$) (si c'est une $\tau$-action). On note $I_2$ l'ensemnle des $(i,j)$ qui sont dans ce cas.
  - Ou bien le lecteur s'est synchronisé avec un autre lecteur, et alors $N_{i,j} = n^\lambda$, et $n$ est un fresh name. : On note $I_3$ l'ensemble des $(i,j)$ dans ce cas.

On fixe $l$ un paramètre de sécurité. La probabilité de succès de l'attaque est :

$$\sum_{i,j} \mathscr{M}_l\left(\llbracket \mathcal{C}_i^j \rrbracket_l\right) \times \left[L_i = N_{i,j} \mid \mathcal{C}_i^j\right]_l$$

Cela est égal à :

$$\sum_{i,j} \mathscr{M}_l\left(\llbracket \mathcal{C}_i^j \wedge L_i = N_{i,j} \rrbracket_l\right)$$

Le but est maintenant d'uttiliser le fait que la distributions sur les names et sur h est uniforme, pour montrer que ces valeurs sont négligeables.

On s'intéresse d'abord à l'ensemble $I_2$ des $i$ et $j$ tels que $N_{i,j} = \text{Choice}\left(N_B^1, split^1\left(h(s', P_{i,j}, c)\right), split^2\left(h(s', P_{i,j}, c)\right)\right)$. Puisque $N_B^1 \notin (P_{i,j}), \left(h(s, N_A^1, M_i)\right), \mathcal{C}_{i,j}$, on peut séparer l'implémentation de $N_B^1$ du reste des termes.

$$\sum_{(i,j)\in I_2} \mathscr{M}_l\left(\llbracket \mathcal{C}_i^j \wedge L_i = N_{i,j} \rrbracket_l\right) = \sum_{\alpha \in \{0,1\}^l} \frac{1}{2^l} \cdot \sum_{(i,j)\in I_2} \mathscr{M}_l\left(\llbracket \mathcal{C}_i^j \wedge s \neq s' \wedge F(\alpha, (h(s, P_{i,j}, c))) = F(\alpha, h(s', P_{i,j}, c)) \rrbracket_l\right)$$
$$+$$
$$\mathscr{M}_l\left(\llbracket \mathcal{C}_i^j \wedge s = s' \rrbracket_l\right)$$

On s'intéresse d'abord au cas où $\eta(s) = \eta(s')$. Comme $s$ et $s'$ sont des names différents, et que la distribution des names est unniforme :

$$\sum_{i,j\in I_2} \mathscr{M}_l\left(\llbracket \mathcal{C}_i^j \wedge s = s' \rrbracket_l\right) \leq \frac{1}{2^l}$$

On note $a = (\mathbf{hash}(s, \mathbf{concat}(P_{i,j}, c)))$, $b = (\mathbf{hash}(s', \mathbf{concat}(P_{i,j}, c)))$, et on constate que :

$$\sum_{i,j\in I_2} \sum_{\alpha \in \{0,1\}^l} \frac{1}{2^l} \cdot \mathscr{M}_l\left(\llbracket \mathcal{C}_i^j \wedge s \neq s' \wedge F(\alpha, \mathbf{hash}(s, \mathbf{concat}(P_{i,j}, c))) = F(\alpha, \mathbf{hash}(s', \mathbf{concat}(P_{i,j}, c))) \rrbracket_l\right)$$
$$= \sum_{i,j\in I_2} \sum_{k\in\{1,..,l\}} (\frac{1}{2})^k \cdot \mathscr{M}_l\left(\llbracket \mathcal{C}_i^j \wedge \forall j\ (a_j = b_j \text{ ou } a_{j+l} = b_{j+l}) \wedge card(t \mid a_t \neq b_t \text{ ou } a_{t+l} \neq b_{t+l}) = k) \rrbracket_l\right)$$

On va maintenant calculer le cardinal des couples de chaines de $\{0,1\}^{2\cdot l}$ qui sont dans l'ensemble :

$$E_i = \{(c,d) \mid \forall j\ (c_j = d_j \text{ ou } c_{j+l} = d_{j+l}) \wedge card(k \mid c_k \neq d_k \text{ ou } c_{k+l} \neq d_{k+l}) = i))$$

Pour choisir un élément de $E_j$, on peut choisir librement $c$, puis choisir les $i$ positions où cela va différer, puis pour chacune de ces positions choisir si cela va différer à droite ou à gauche. On a (avec $C_n^i$ le coefficient binomial) :

$$Card(E_i) = 2^{2l}2^i \cdot C_l^i$$

On va maintenant appliquer le lemme 4 : On s'intéresse à la complexité syntaxique des contraintes et des termes considéré : elle est borné par :

$$card(\text{termes qui interviennent dans une éxécution}) \times (\text{longueur syntaxique d'un terme}) \leq 2 \cdot P(l) \times P(l)$$

.

On peut déduire du lemme 4 qu'il existe un polynome R tel que :

$$\sum_{i,j|\text{ l'adversaire a choisi une } \tau \text{ action}} \mathscr{D}_l(\eta \mid \eta(\mathcal{C}_i^j)\text{est vrai}) \times \left[L_i = N_{i,j} \mid \mathcal{C}_i^j\right]_l \leq (R(l))^2 \cdot \sum_{j\in\{1,\cdots,l\}} \frac{2^j C_l^j}{2^j \times 2^{2l}} \leq \frac{R(l)^2}{2^l}$$

On s'intéresse maintenant au cas où $N_B^1 \notin Names(N_{i,j})$. On peut donc séparer l'implémentation de $N_B^1$ du reste des termes. On définit $\mathscr{E}_l$ par : $\mathscr{D}_l = \sum_{\alpha\in\{0,1\}^l} \frac{1}{2^l}\delta_\alpha \cdot \mathscr{E}_l$

$$\begin{aligned}
&\mathscr{D}_l\left(\eta \mid \eta(\mathcal{C}_i^j) \text{ est vrai} \wedge \eta(L_i) = \eta(N_{i,j})\right) \\
&= \sum_{\alpha\in\{0,1\}^l} \frac{1}{2^l}\mathscr{E}_l(\eta \mid \eta(\mathcal{C}_i^j) \text{ est vrai} \\
&\wedge \eta(\text{Choice}\left(\alpha, split^1\left(h(s, P_{i,j}, c)\right), split^2\left(h(s, P_{i,j}, c)\right)\right)) = \eta(N_{i,j}))
\end{aligned}$$

On note $a = fst\left(h(s, P_{i,j}, c)\right)$, $b = snd\left(h(s, P_{i,j}, c)\right)$ .

$$= \sum_{i\in\{1,..,l\}} (\frac{1}{2})^i \mathscr{E}_l(\eta \mid \eta(\mathcal{C}_i^j) \text{ est vrai } \wedge card\{k \mid \eta(a)_k \neq \eta(b)_k\} = i)$$

On va maintenant calculer le cardinal des chaines de $\{0,1\}^{2\cdot l}$ qui sont dans l'ensemble :

$$A_j = \{c \mid card\{k \mid \eta(fst(a))_k \neq \eta(snd(a))_k\} = j\}\}$$

On a (avec $C_n^i$ le coefficient binomial) :

$$Card(A_j) = 2^l \cdot C_l^j$$

On déduit du lemme 3 :

$$\sum_{i,j|\text{ l'adversaire a choisi une fast}(\cdot)(N) \text{ action}} \mathscr{D}_l(\eta \mid \eta(\mathcal{C}_i^j)\text{est vrai}) \times \left[L_i = N_{i,j} \mid \mathcal{C}_i^j\right]_l \leq (P(l)) \cdot \sum_{j\in\{1,\cdots,l\}} \frac{2^l C_l^j}{2^{2l}} \leq P(l) \cdot \left(\frac{3}{4}\right)^l$$

En réunissant les deux, on obtient :

$$\text{Prob de succès de l'adversaire} \leq Q(l)\frac{3}{4}^l$$

où Q est un polynome.

# J Bayesian Authentication (C.Meadows/D.Pavlovic) : Counterexample to the proposition 4.7 de [11]

On définit une fonction sur les bistrings :

$$\text{parity}(x) = x \text{ if } \sum_i x_i = 0 \bmod 2, \text{ not } (x) \text{ otherwise.}$$

Le but est d'avoir une fonction telle que $x = parity(x)$ avec probabilité $\frac{1}{2}$. Alors :

$$\begin{aligned} P \stackrel{\text{def}}{=} & \nu x, M, N \cdot \\ & \text{let} L = \text{ if } (\text{parity}(x) = x) \text{ then } M \text{ else } N \\ & \text{send } M \\ & \cdot \text{ receive } K \\ & \cdot \text{ if } (N = K) \text{ then } Ok \text{ else } Fail \end{aligned}$$

et $L, M, N$ sont des nonces de longueur $l$.

Alors dans la moitié des cas (quand $x = parity(x)$), le terme $L$ est égal à $M$, et dans l'autre moitié des cas, le terme $L$ est égal à $N$.

$Pr(M \vdash N) = \frac{1}{2^l}$.
et de plus : $Pr(M \vdash N)_l \geq Pr(M \vdash N, L)$, so $Pr(M \vdash N, L) \leq \frac{1}{2^l}$ Mais si nous appliquons la proposition 4.7 : $Pr(M \vdash N, L) \geq Pr(M \vdash L) \cdot Pr(M, L \vdash N)$.
Pour deviner $L$ à partir $M$, l'adversaire peut simplement renvoyer $M$, et il a la bonne chaîne de caractère avec probabilité $\frac{1}{2}$. Ainsi $Pr(M \vdash L) \geq \frac{1}{2}$.
De même, pour deviner $N$ à partie de $M$ et de $L$, l'adversaire peut simplement renvoyer $L$, et il a la bonne chaîne de caractère avec probabilité $\frac{1}{2}$. Ainsi $Pr(M, L \vdash N) \geq \frac{1}{2}$.
Ainsi on devrait avoir en appliquant la proposition 4.7 : $Pr(M \vdash N) \geq \frac{1}{4}$, ce qui est faux.