Spi-calcul et classes $\mathcal{H}_1$ et $\mathcal{H}_2$

Correction.

Rappel de la première partie (que vous avez dû faire à la maison.)

On considère des programmes écrit dans le *spi-calcul*. D'abord les expressions :

$$\begin{array}{rcll} e, \ldots & ::= & x & \text{variables} \\ & | & \{e_1\}_{e_2} & \text{chiffrements} \\ & | & \langle e_1, e_2 \rangle & \text{paires} \end{array}$$

Ces expressions dénotent des valeurs. Ensuite, les processus :

$$\begin{array}{rcll} P, Q, R, \ldots & ::= & 0 & \text{stop} \\ & | & !P & \text{réplication} \\ & | & P|Q & \text{parallèle} \\ & | & \text{case } e_1 \text{ of } \{x\}_{e_2} \Rightarrow Q & \text{déchiffrement} \\ & | & \text{case } e_1 \text{ of } \langle x, y \rangle \Rightarrow Q & \text{extraction de composantes} \\ & | & [e_1 = e_2]P & \text{test d'égalité} \\ & | & (\nu x)P & \text{création de nom frais} \\ & | & e_1\langle e_2 \rangle.P & \text{émission} \\ & | & e_1(x).P & \text{réception} \end{array}$$

Étant donné un processus P, l'ensemble $\text{sp}(P)$ des *sous-processus* de P et l'ensemble $\text{fv}(P)$ des variables libres de P sont définis comme suit :

$$\begin{array}{rclrcl} \text{sp}(0) & = & \emptyset & \text{fv}(0) & = & \emptyset \\ \text{sp}(!P) & = & \text{sp}(P) \cup \{!P\} & \text{fv}(!P) & = & \text{fv}(P) \\ \text{sp}(P|Q) & = & \text{sp}(P) \cup \text{sp}(Q) \cup \{P|Q\} & \text{fv}(P|Q) & = & \text{fv}(P) \cup \text{fv}(Q) \\ \text{sp}((\nu x)P) & = & \text{sp}(P) \cup \{(\nu x)P\} & \text{fv}((\nu x)P) & = & \text{fv}(P) \setminus \{x\} \\ \text{sp}(e_1\langle e_2\rangle.P) & = & \text{sp}(P) \cup \{e_1\langle e_2\rangle.P\} & \text{fv}(e_1\langle e_2\rangle.P) & = & \text{fv}(e_1) \cup \text{fv}(e_2) \cup \text{fv}(P) \\ \text{sp}(e_1(x).P) & = & \text{sp}(P) \cup \{e_1(x).P\} & \text{fv}(e_1(x).P) & = & \text{fv}(e_1) \cup (\text{fv}(P) \setminus \{x\}) \\ \text{sp}([e_1 = e_2]P) & = & \text{sp}(P) \cup \{[e_1 = e_2]P\} & \text{fv}([e_1 = e_2]P) & = & \text{fv}(e_1) \cup \text{fv}(e_2) \cup \text{fv}(P) \end{array}$$

si $P = \text{case } e_1 \text{ of } \{x\}_{e_2} \Rightarrow Q$:

$$\begin{array}{rclrcl} \text{sp}(P) & = & \text{sp}(Q) \cup \{P\} & \text{fv}(P) & = & \text{fv}(e_1) \cup \text{fv}(e_2) \cup (\text{fv}(Q) \setminus \{x\}) \end{array}$$

si $P = \text{case } e_1 \text{ of } \langle x, y \rangle \Rightarrow Q$:

$$\begin{array}{rclrcl} \text{sp}(P) & = & \text{sp}(Q) \cup \{P\} & \text{fv}(P) & = & \text{fv}(e_1) \cup (\text{fv}(Q) \setminus \{x, y\}) \end{array}$$

où l'ensemble $\mathrm{fv}(e)$ des variables libres d'une expression e est défini par $\mathrm{fv}(x) = \{x\}$, $\mathrm{fv}(\{e_1\}_{e_2}) = \mathrm{fv}(e_1) \cup \mathrm{fv}(e_2)$, $\mathrm{fv}(\langle e_1, e_2\rangle) = \mathrm{fv}(e_1) \cup \mathrm{fv}(e_2)$. On ne définira pas formellement l'ensemble des sous-expressions de P, qui est défini de façon analogue à l'ensemble des sous-processus.

On notera que $\mathrm{sp}(P)$ est toujours un ensemble fini de processus. Fixons un processus donné P_0 clos, c'est-à-dire tel que $\mathrm{fv}(P_0) = \emptyset$. Pour chaque processus de la forme $(\nu x)Q$ dans $\mathrm{sp}(P_0)$, ayant k variables libres $x_1, \ldots, x_k$, soit $n_{(\nu x)Q}$ un symbole de fonction d'arité k. L'idée est que le terme $n_{(\nu x)Q}(x_1, \ldots, x_k)$ dénote la valeur du nonce frais x lorsqu'on exécute $(\nu x)Q$, connaissant les valeurs $x_1, \ldots, x_k$.

On définit une sémantique simplifiée du spi-calcul comme suit. Pour tout $P \in \mathrm{sp}(P_0)$, on définit un jugement $\vdash \mathrm{acc}_P$ ("l'exécution de P_0 peut mener à devoir exécuter P") ; pour toute sous-expression e de P_0, on définit un second jugement $\vdash e = t$ ("il se peut que e vaille le terme t" — on notera qu'il n'y a qu'un nombre fini de sous-expressions e) ; finalement, on définit un jugement $t \triangleright t'$ ("sur le canal t, on peut voir passer le message t'").

Les règles permettant de dériver ces jugements sont comme suit. D'abord, P_0 peut démarrer :

$$\frac{}{\vdash P_0}\,(Start)$$

On considère ensuite chaque construction possible de processus :

$$\frac{\vdash !P}{\vdash P}\,(!) \quad \frac{\vdash P|Q}{\vdash P}\,(|_1) \quad \frac{\vdash P|Q}{\vdash Q}\,(|_2) \quad \frac{\vdash (\nu x)P}{\vdash P}\,(\nu)$$

$$\frac{\vdash (\nu x)P \quad \vdash x_1 = t_1 \ldots \vdash x_k = t_k}{\vdash x = n_{(\nu x)P}(t_1, \ldots, t_k)}\,(x = \nu)$$
$$(\text{où } \mathrm{fv}((\nu x)P) = \{x_1, \ldots, x_k\})$$

$$\frac{\vdash \text{case } e_1 \text{ of } \{x\}_{e_2} \Rightarrow Q \quad \vdash e_1 = \{t\}_{e_2[x_1:=t_1,\ldots,x_k:=t_k]} \quad \vdash x_1 = t_1 \ldots \vdash x_k = t_k}{\vdash Q}\,(dec)$$

$$\frac{\vdash \text{case } e_1 \text{ of } \{x\}_{e_2} \Rightarrow Q \quad \vdash e_1 = \{t\}_{e_2[x_1:=t_1,\ldots,x_k:=t_k]} \quad \vdash x_1 = t_1 \ldots \vdash x_k = t_k}{\vdash x = t}\,(x = dec)$$

où $x_1, \ldots, x_k$ sont les variables libres de e_2.

$$\frac{\vdash \text{case } e_1 \text{ of } \langle x, y\rangle \Rightarrow Q \quad \vdash e_1 = \langle t, u\rangle}{\vdash Q}\,(proj)$$

$$\frac{\vdash \text{case } e_1 \text{ of } \langle x, y\rangle \Rightarrow Q \quad \vdash e_1 = \langle t, u\rangle}{\vdash x = t}\,(x = \pi_1) \qquad \frac{\vdash \text{case } e_1 \text{ of } \langle x, y\rangle \Rightarrow Q \quad \vdash e_1 = \langle t, u\rangle}{\vdash y = u}\,(x = \pi_2)$$

$$\frac{\vdash [e_1 = e_2]P \quad \vdash e_1 = t \quad \vdash e_2 = t}{\vdash P}\,(=)$$

On a besoin ici de jugements auxiliaires exprimant les valeurs possibles d'expressions comme e_1 ou e_2 en fonction des valeurs des variables qui y sont libres :

$$\frac{\vdash e_1 = t_1 \quad \vdash e_2 = t_2}{\vdash \{e_1\}_{e_2} = \{t_1\}_{t_2}}\,(crypt) \qquad \frac{\vdash e_1 = t_1 \quad \vdash e_2 = t_2}{\vdash \langle e_1, e_2\rangle = \langle t_1, t_2\rangle}\,(pair)$$

L'émission de messages permet de définir une autre forme de jugement, $t \rhd t'$, qui exprime que le canal de communication t peut voir passer le message t' :

$$\frac{\vdash e_1\langle e_2\rangle.P}{\vdash P}\,(send) \qquad \frac{\vdash e_1\langle e_2\rangle.P \quad \vdash e_1 = t_1 \quad \vdash e_2 = t_2}{t_1 \rhd t_2}\,(\rhd)$$

Ce nouveau jugement permet de définir les messages possiblement reçus par une instruction de réception :

$$\frac{\vdash e_1(x).P \quad \vdash e_1 = t_1 \quad t_1 \rhd t}{\vdash P}\,(recv) \qquad \frac{\vdash e_1(x).P \quad \vdash e_1 = t_1 \quad t_1 \rhd t}{\vdash x = t}\,(x = recv)$$

1. On demande de coder les jugements définis ci-dessus sous forme de clauses de Horn. On rappelle que P_0 est un processus supposé fixé, que $\mathrm{sp}(P_0)$ est fini et qu'il n'y a qu'un nombre fini de sous-expressions de P_0. On utilisera, pour chaque sous-processus $P \in \mathrm{sp}(P_0)$ de variables libres $x_1, \ldots, x_k$, un prédicat acc_P tel que $\mathrm{acc}_P(*)$ exprime $\vdash P$ (où $*$ est une constante) ; un prédicat $\mathrm{val}\langle t, t'\rangle$ exprimant que le canal t peut voir passer le message t', c'est-à-dire $t \rhd t'$; et pour chaque sous-expression e de P_0 un prédicat $\mathrm{eq}_e(t)$ exprimant $\vdash e = t$.

   Formellement, on traduira les règles $(Start)$, $(!)$, $(|_1)$, $(|_2)$, (ν), $(x = \nu)$, (dec), $(x = dec)$, $(proj)$, $(x = \pi_1)$, $(x = \pi_2)$, $(=)$, $(crypt)$, $(pair)$, $(send)$, $(\rhd)$, $(recv)$, $(x = recv)$, en des ensembles de clauses définies dont le plus petit modèle vérifie que :

   (i) $\mathrm{acc}_P(u)$ y est vrai si et seulement si $\vdash P$ est dérivable et $u = *$;

   (ii) $\mathrm{val}(u)$ est vrai si et seulement si u est de la forme $\langle t, t'\rangle$, et $t \rhd t'$;

   (iii) $\mathrm{eq}_e(u)$ y est vrai si et seulement si $\vdash e = u$ est dérivable.

   On supposera pour simplifier, et ce sans perte de généralité, que toutes les variables liées dans chaque processus sont deux à deux disjointes.

   Les clauses obtenues sont elles dans la classe $\mathcal{H}_1$? On rappelle que le classe $\mathcal{H}_1$ est la classe des clauses de Horn de l'une des formes :

$$\begin{array}{rcl} \bot & \Leftarrow & P_1(t_1), \ldots, P_k(t_k) \\ P(X) & \Leftarrow & P_1(t_1), \ldots, P_k(t_k) \\ P(f(X_1, \ldots, X_n)) & \Leftarrow & P_1(t_1), \ldots, P_k(t_k) \end{array}$$

   où, dans le dernier cas, les variables $X_1, \ldots, X_n$ sont deux à deux disjointes.

C'est immédiat :
- $(Start)$ $\mathrm{acc}_{P_0}(*)$.
- $(!)$ $\mathrm{acc}_P(*) \Leftarrow \mathrm{acc}_{!P}(*)$.
- $(|_1)$ $\mathrm{acc}_P(*) \Leftarrow \mathrm{acc}_{P|Q}(*)$.
- $(|_2)$ $\mathrm{acc}_Q(*) \Leftarrow \mathrm{acc}_{P|Q}(*)$.
- (ν) $\mathrm{acc}_P(*) \Leftarrow \mathrm{acc}_{(\nu x)P}(*)$.
- $(x = \nu)$ $\mathrm{eq}_x(n_{(\nu x)P}(X_1, \ldots, X_k)) \Leftarrow \mathrm{acc}_{(\nu x)P}(*), \mathrm{eq}_{x_1}(X_1), \ldots, \mathrm{eq}_{x_k}(X_k)$*, où les variables libres de* $(\nu x)P$ *sont supposées être* $x_1, \ldots, x_k$*, et* X_1*, …,* X_k *sont* k *variables du premier ordre fraîches.*
- (dec) $\mathrm{acc}_Q(*) \Leftarrow \mathrm{acc}_{\mathrm{case}\ e_1\ \mathrm{of}\ \{x\}_{e_2} \Rightarrow Q}(*), \mathrm{eq}_{e_1}(\{X\}_{e_2[x_1:=X_1,\ldots,x_k:=X_k]}), \mathrm{eq}_{x_1}(X_1), \ldots, \mathrm{eq}_{x_k}(X_k)$.
- $(x = dec)$ $\mathrm{eq}_x(X) \Leftarrow \mathrm{acc}_{\mathrm{case}\ e_1\ \mathrm{of}\ \{x\}_{e_2} \Rightarrow Q}(*), \mathrm{eq}_{e_1}(\{X\}_{e_2[x_1:=X_1,\ldots,x_k:=X_k]}), \mathrm{eq}_{x_1}(X_1), \ldots, \mathrm{eq}_{x_k}(X_k)$
- $(proj)$ $\mathrm{acc}_Q(*) \Leftarrow \mathrm{acc}_{\mathrm{case}\ e_1\ \mathrm{of}\ \langle x,y\rangle \Rightarrow Q}(*), \mathrm{eq}_{e_1}(\langle X, Y\rangle)$.
- $(x = \pi_1)$ $\mathrm{eq}_x(X) \Leftarrow \mathrm{acc}_{\mathrm{case}\ e_1\ \mathrm{of}\ \langle x,y\rangle \Rightarrow Q}(*), \mathrm{eq}_{e_1}(\langle X, Y\rangle)$.
- $(x = \pi_1)$ $\mathrm{eq}_y(Y) \Leftarrow \mathrm{acc}_{\mathrm{case}\ e_1\ \mathrm{of}\ \langle x,y\rangle \Rightarrow Q}(*), \mathrm{eq}_{e_1}(\langle X, Y\rangle)$.
- $(=)$ $\mathrm{acc}_P(*) \Leftarrow \mathrm{acc}_{[e_1=e_2]P}(*), \mathrm{eq}_{e_1}(X), \mathrm{eq}_{e_2}(X)$.
- $(crypt)$ $\mathrm{eq}_{\{e_1\}_{e_2}}(\{X\}_Y) \Leftarrow \mathrm{eq}_{e_1}(X), \mathrm{eq}_{e_2}(Y)$.
- $(pair)$ $\mathrm{eq}_{\langle e_1,e_2\rangle}(\langle X, Y\rangle) \Leftarrow \mathrm{eq}_{e_1}(X), \mathrm{eq}_{e_2}(Y)$.
- $(send)$ $\mathrm{acc}_P(*) \Leftarrow \mathrm{acc}_{e_1\langle e_2\rangle.P}(*)$.
- $(\triangleright)$ $\mathrm{val}\langle X, Y\rangle \Leftarrow \mathrm{acc}_{e_1\langle e_2\rangle.P}(*), \mathrm{eq}_{e_1}(X), \mathrm{eq}_{e_2}(Y)$.
- $(recv)$ $\mathrm{acc}_P(*) \Leftarrow \mathrm{acc}_{e_1(x).P}(*), \mathrm{eq}_{e_1}(X), \mathrm{val}\langle X, Y\rangle$.
- $(x = recv)$ $\mathrm{eq}_x(Y) \Leftarrow \mathrm{acc}_{e_1(x).P}(*), \mathrm{eq}_{e_1}(X), \mathrm{val}\langle X, Y\rangle$.

Elles sont toutes dans $\mathcal{H}_1$*. En fait, la forme des règles a été étudiée de sorte que ce résultat soit atteint.*

2. Si on avait remplacé les règles $(crypt)$ et $(pair)$ par l'unique règle :

$$\frac{\vdash x_1 = t_1 \quad \ldots \quad \vdash x_k = t_k}{\vdash e = e[x_1 := t_1, \ldots, x_k := t_k]}\ (expr)$$

montrer que les clauses obtenues ne seraient pas tombées dans $\mathcal{H}_1$ en général.

La traduction aurait été $\mathrm{eq}_e(e[x_1 := X_1, \ldots, x_k := X_k]) \Leftarrow \mathrm{eq}_{x_1}(X_1), \ldots, \mathrm{eq}_{x_k}(X_k)$*, qui n'aurait pas été dans* $\mathcal{H}_1$ *dès que* e *était non linéaire, par exemple.*

3. En conclure par les résultats du cours (à l'oral, ou bien d'après les résultats de l'examen 2003), que l'on peut vérifier si $\vdash P$, pour tout $P \in \mathrm{sp}(P_0)$, en temps exponentiel en la taille de P_0. De même pour vérifier toute propriété de la forme $\exists X_1, \ldots, X_k \cdot F$, où F est une conjonction de formules atomiques dépendant de X_1, …, X_k, construites sur les prédicats acc_P, eq_e, val.

Clairement la taille de l'ensemble de clauses produit en question 1 est polynomial, en fait linéaire, en la taille de P_0*. On utilise ensuite que décider si cet ensemble de clauses, plus la clause* $\perp \Leftarrow \mathrm{acc}_P(*)$ *(qui est dans* $\mathcal{H}_1$*), se fait en temps exponentiel.*

4. On considère les processus suivants, représentant le protocole de Needham-Schroeder à clés symétriques :

1.	$A \longrightarrow S : A, B, N_a$
2.	$S \longrightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$
3.	$A \longrightarrow B : \{K_{ab}, A\}_{K_{bs}}$
4.	$B \longrightarrow A : \{N_b\}_{K_{ab}}$
5.	$A \longrightarrow B : \{N_b, B\}_{K_{ab}}$

$$
\begin{array}{rcl}
A(c_{\to s}, c_{\leftarrow s}, c_{\to b}, c_{\leftarrow b}, a, b, k_{as}) & =_{def} & (\nu N_a) c_{\to s}\langle\langle a, \langle b, N_a\rangle\rangle\rangle . c_{\leftarrow s}(x_1). \\
& & \quad \text{case } x_1 \text{ of } \{x_2\}_{k_{as}} \Rightarrow \\
& & \quad \text{case } x_2 \text{ of } \langle N'_a, x_3\rangle \Rightarrow \\
& & \quad [N_a = N'_a]\text{case } x_3 \text{ of } \langle B, x_4\rangle \Rightarrow \\
& & \quad [b = B]\text{case } x_4 \text{ of } \langle k^a_{ab}, x_5\rangle \Rightarrow \\
& & \quad c_{\to b}\langle x_5\rangle . c_{\leftarrow b}(x_6). \\
& & \quad \text{case } x_6 \text{ of } \{x_7\}_{k^a_{ab}} \Rightarrow \\
& & \quad c_{\to b}\langle\langle x_7, b\rangle\rangle .0 \\
S(c_{\leftarrow a}, c_{\to a}, k_{as}, k_{bs}) & =_{def} & c_{\leftarrow a}(y_1). \\
& & \quad \text{case } y_1 \text{ of } \langle y_2, y_3\rangle \Rightarrow \\
& & \quad \text{case } y_3 \text{ of } \langle y_4, y_5\rangle \Rightarrow \\
& & \quad (\nu k^s_{ab}) c_{\to a}\langle\{\langle y_5, \langle y_4, \langle k^s_{ab}, \{\langle k^s_{ab}, y_2\rangle\}_{k_{bs}}\rangle\rangle\rangle\}_{k_{as}}\rangle .0 \\
B(c_{\leftarrow a}, c_{\to a}, k_{bs}) & =_{def} & c_{\leftarrow a}(z_1). \\
& & \quad \text{case } z_1 \text{ of } \{z_2\}_{k_{bs}} \Rightarrow \\
& & \quad \text{case } z_2 \text{ of } \langle k^b_{ab}, z_3\rangle \Rightarrow \\
& & \quad (\nu N_b) c_{\to a}\langle\{N_b\}_{k^b_{ab}}\rangle . \\
& & \quad c_{\leftarrow a}(z_4).\text{case } z_4 \text{ of } \{z_5\}_{k^b_{ab}} \Rightarrow \\
& & \quad \text{case } z_5 \text{ of } \langle z_6, z_7\rangle \Rightarrow \\
& & \quad [z_6 = N_b]0 \\
I(c_{pub}, a, b) & =_{def} & (c_{pub}(m).c_{pub}(k).c_{pub}\langle\{m\}_k\rangle .0) \\
& | & (c_{pub}(m).c_{pub}(k).\text{case } m \text{ of } \{x\}_k \Rightarrow c_{pub}\langle x\rangle .0) \\
& | & (c_{pub}(m).c_{pub}(m').c_{pub}\langle\langle m, m'\rangle\rangle .0) \\
& | & (c_{pub}(m).\text{case } m \text{ of } \langle x, y\rangle \Rightarrow c_{pub}\langle x\rangle .0) \\
& | & (c_{pub}(m).\text{case } m \text{ of } \langle x, y\rangle \Rightarrow c_{pub}\langle y\rangle .0) \\
& | & c_{pub}\langle a\rangle \mid c_{pub}\langle b\rangle \\
\\
Sys & =_{def} & (\nu c_{pub})(\nu a)(\nu b)(\nu k_{as})(\nu k_{bs}) \\
& & \quad !A(c_{pub}, c_{pub}, c_{pub}, c_{pub}, a, b, k_{as}) \\
& & \quad \mid\ !B(c_{pub}, c_{pub}, k_{bs}) \\
& & \quad \mid\ !S(c_{pub}, c_{pub}, k_{as}, k_{bs}) \\
& & \quad \mid\ !I(c_{pub}, a, b)
\end{array}
$$

L'intrus de Dolev-Yao y est représenté par le processus I, et le protocole y est représenté par le terme clos Sys.

Quelle propriété doit-on écrire pour décrire que la clé k^a_{ab} reçue par A reste secrète ? pour

k^b_{ab} ? pour k^s_{ab} ? Comment s'expriment-elles sous forme de clauses ?

La clé k^a_{ab} reçue par A est possiblement connu de l'intrus si et seulement si les jugements $\vdash k^a_{ab} = t$, $\vdash c_{pub} = u$ et $u \triangleright t$ sont dérivables pour certains termes t et u. Le secret de k^a_{ab} se code donc par la clause

$$\perp \Leftarrow \mathrm{eq}_{k^a_{ab}}(X), \mathrm{eq}_{c_{pub}}(Y), \mathrm{val}\langle Y, X\rangle$$

De même pour k^b_{ab} :

$$\perp \Leftarrow \mathrm{eq}_{k^b_{ab}}(X), \mathrm{eq}_{c_{pub}}(Y), \mathrm{val}\langle Y, X\rangle$$

et pour k^s_{ab} :

$$\perp \Leftarrow \mathrm{eq}_{k^s_{ab}}(X), \mathrm{eq}_{c_{pub}}(Y), \mathrm{val}\langle Y, X\rangle$$

5. La classe $\mathcal{H}_2^k$ ($k \geq 1$) est définie comme la classe des clauses de $\mathcal{H}_1$ vérifiant de plus les propriétés :

   (a) les variables libres dans la tête apparaissent au plus une fois dans le corps de la clause ;

   (b) les variables libres du corps ont au plus k occurrences.

   Par exemple, $P(f(X_1, X_2)) \Leftarrow Q(X_1), R(g(X_2, X_3, X_3))$ est une clause de $\mathcal{H}_2^2$. (NB : comme on va utiliser la règle de splitting sans splitting, nous étendons ici légèrement la définition des clauses de $\mathcal{H}_1$, qui seront toutes clauses de la forme $A \Leftarrow P_1(t_1), \ldots, P_m(t_m), q_1, \ldots, q_j$, où A vaut $\perp$, est de la forme $P(X)$, ou bien $P(f(X_1, \ldots, X_n))$, les variables X_i étant deux à deux distinctes, ou bien de la forme q ; q, q_1, …, q_j sont des *littéraux de splitting*, c'est-à-dire des prédicats 0-aires de la forme $\ulcorner \perp \Leftarrow B'(Y) \urcorner$ où B' est un bloc non vide $\{P'_1, \ldots, P'_m\}$, la sémantique intuitive d'un tel littéral de splitting étant d'exprimer que l'intersection des langages reconnus en P'_1, …, P'_m est non vide.)

   On rappelle que la stratégie de résolution ordonnée avec sélection, avec splitting sans splitting, pour la classe $\mathcal{H}_1$, revenait à utiliser les règles suivantes :

   – Résolution (1) :

   $$\frac{P(f(X_1, \ldots, X_n)) \Leftarrow B_1(X_1), \ldots, B_n(X_n) \quad A \Leftarrow P(f(t_1, \ldots, t_n)), P_1(t'_1), \ldots, P_m(t'_m)}{A \Leftarrow B_1(t_1), \ldots, B_n(t_n), P_1(t'_1), \ldots, P_m(t'_m)}$$

   où B_1, …, B_n, B désignent des blocs (possiblement vides), c'est-à-dire des ensembles de prédicats (par exemple, si $B_1 = \{Q, R\}$, $B_1(X_1)$ est la conjonction $Q(X_1), R(X_1)$) ; et A est $\perp$, ou un atome $Q(t)$, ou un littéral de splitting q ;

   – Résolution (2) :

   $$\frac{P(f(X_1, \ldots, X_n)) \Leftarrow B_1(X_1), \ldots, B_n(X_n) \quad A \Leftarrow P(X), B(X)}{A[X := f(X_1, \ldots, X_n)] \Leftarrow B_1(X_1), \ldots, B_n(X_n), B(f(X_1, \ldots, X_n))}$$

   où B_1, …, B_n désignent des blocs (possiblement vides) ; A est $\perp$, ou de la forme $Q(X)$, ou bien un littéral de splitting q

– Résolution (3) :

$$\frac{q \qquad A \Leftarrow P_1(t_1), \ldots, P_m(t_m), q, q_1, \ldots, q_j}{A \Leftarrow P_1(t_1), \ldots, P_m(t_m), q_1, \ldots, q_j}$$

où A est $\bot$, ou de la forme $Q(X)$, ou bien un littéral de splitting, et q, q_1, …, q_j sont des littéraux de splitting ; (note : nous n'avons pas évoqué cette règle en cours, mais c'est l'un des cas qui se présentent en présence de littéraux de splitting ;)

– Splitting sans splitting :

$$\frac{A \Leftarrow P_1(t_1), \ldots, P_m(t_m), B(X), q_1, \ldots, q_j}{\begin{array}{c} A \Leftarrow P_1(t_1), \ldots, P_m(t_m), q, q_1, \ldots, q_j \\ q \Leftarrow B(X) \end{array}}$$

où $q = \ulcorner \bot \Leftarrow B(X) \urcorner$; A vaut $\bot$, un littéral de splitting q, ou un atome $P(t)$; q_1, …, q_j sont des littéraux de splitting ; B est un bloc non vide ; et X n'est pas libre ni dans A ni dans aucun des t_i, $1 \leq i \leq m$.

On supposera toujours que la règle de splitting sans splitting est appliquée en priorité. Formellement, ceci signifie que l'on suppose que les trois règles de résolution ne sont appliquées que sur des prémisses qui ne sont pas des prémisses de la règle de splitting sans splitting.

Montrer que si l'on applique ces règles à des clauses de $\mathcal{H}_2^k$, on obtient encore des clauses de $\mathcal{H}_2^k$.

Dans la règle de résolution (1) et dans la règle de résolution (2), le fait que la prémisse de gauche soit dans $\mathcal{H}_2^k$ implique que les blocs B_i, $1 \leq i \leq n$, contiennent au plus un symbole de prédicat.

Dans la règle de résolution (1), si X est libre dans la tête A de la conclusion, alors comme la prémisse de droite est dans $\mathcal{H}_2^k$, X est libre soit dans exactement un des t_i, soit dans exactement un des t'_i. Dans le premier cas, X apparaît au plus une fois dans le corps de la conclusion car B_i contient au plus un prédicat. Dans le second cas, X apparaît exactement une fois dans le corps de la conclusion, dans t'_i. La condition **(a)** *est donc vraie dans la conclusion.*

Dans la règle de résolution (2), toute variable libre dans la tête de la conclusion est nécessairement parmi X_1, …, X_n, et A doit alors être de la forme $Q(X)$. La variable X_i apparaît alors au plus une fois dans la partie $B_1(X_1), \ldots, B_n(X_n)$ du corps, car B_i contient au plus un prédicat. D'autre part, comme X est libre dans A et que la prémisse de droite est dans $\mathcal{H}_2^k$, c'est que B est vide. Donc X_i apparaît au plus une fois dans le corps de la conclusion. La condition **(a)** *est encore vérifiée de la conclusion.*

La condition **(a)** *est triviale pour la règle de résolution (3). Pour la règle de splitting sans splitting, elle est triviale aussi.*

Vérifions la condition **(b)**. *Comme $k \geq 1$, il suffit de vérifier que toute variable libre dans le corps mais pas dans la tête apparaît au plus k fois.*

Dans le cas de la règle de résolution (1), si X apparaît libre dans le corps de la conclusion mais n'est pas libre dans la tête A, c'est que X apparaît au plus k fois dans le corps $P(f(t_1, \ldots, t_n)), P_1(t'_1), \ldots, P'_m(t'_m)$ de la prémisse de droite. Comme chaque B_i contient au plus un prédicat, X ne peut apparaître qu'au plus k fois dans le corps de la conclusion.

Dans le cas de la règle de résolution (2), il n'existe de variable dans le corps de la conclusion qui ne soit pas libre dans la tête que si A est $\perp$ ou un littéral de splitting q. Il s'agit alors d'une variable X_i. Comme B est vide, X_i apparaît au plus 1 fois, donc au plus k fois puisque $k \geq 1$.

Dans le cas de la règle de résolution (3), c'est évident. De même pour la règle de splitting sans splitting.

6. Intuitivement, lorsque B n'est pas vide, la seule chose que l'on peut faire avec la conclusion d'une règle de résolution (2), c'est d'unifier les littéraux restant de $B(f(X_1, \ldots, X_n))$ avec d'autres clauses automates de tête $P(f(X_1, \ldots, X_n))$. On admettra donc que la procédure reste complète si l'on remplace la règle de résolution (2) par la règle dite de résolution (2') :

$$\frac{\overbrace{P_j(f(X_1, \ldots, X_n)) \Leftarrow B_{j1}(X_1), \ldots, B_{jn}(X_n)}^{1 \leq j \leq m} \quad A \Leftarrow P_1(X), \ldots, P_m(X)}{A[X := f(X_1, \ldots, X_n)] \Leftarrow B_{11}(X_1), \ldots, B_{m1}(X_1), \ldots, B_{1n}(X_n), \ldots, B_{mn}(X_n)}$$

Par la question précédente, on notera que soit A est $\perp$ ou un littéral de splitting, et chaque X_i apparaît au plus k fois (et la conclusion se splitte aussitôt), soit A est de la forme $Q(X)$, et chaque X_i apparaît au plus une fois dans le corps de la conclusion.

On définit la *taille* des termes, des atomes, et des corps de clauses par : $|X| = 1$, $|f(t_1, \ldots, t_n)| = 1 + \sum_{i=1}^{n} |t_i|$, $|P(t)| = |t|$, $|q| = 1$, et la taille d'un corps de clause est la somme des tailles de ses atomes.

Montrer que si S est un ensemble de clauses de $\mathcal{H}_2^k$, dont la taille des corps est bornée par s, et n'utilisant que des symboles de fonction d'arité au plus a, alors toutes les clauses produites ont des corps de taille au plus $\max(k(a + 1), s)$.

Montrer ensuite que les termes t arguments de prédicats dans les clauses C produites par ces règles à partir de S sont soit (a) des sous-termes de termes présents dans S, soit (b) des variables. On notera N le nombre de sous-termes de termes présents dans S.

En déduire que si S ne contient pas de symbole de splitting, alors l'application exhaustive de ces règles ne produit qu'un nombre *polynomial* de clauses de $\mathcal{H}_2^k$, à k, s, et a fixés. En déduire un algorithme de décision de S en temps polynomial, à k, s, et a fixés.

- *la règle de résolution (1) fabrique une conclusion dont le corps est de taille inférieure ou égale à celle du corps de la prémisse de droite, puisque chaque B_i contient au plus un prédicat, et se splitte en une clause dont le corps n'est pas plus gros, donc de taille au plus* $\max(k(a + 1), s)$ *par hypothèse de récurrence ;*

- *la règle de résolution (2), donc aussi (2'), fabrique une conclusion dont le corps est de taille au plus n plus le cardinal de B fois $n+1$, soit au plus $a+(k-1)(a+1) = ka+k-1 \leq k(a+1)$;*
- *La règle de résolution (3) diminue la taille du corps de la conclusion par rapport à la prémisse de droite, et diminue le nombre de littéraux de splitting présents ;*
- *la règle de splitting sans splitting fabrique une première conclusion de taille inférieure ou égale, et une deuxième conclusion dont le corps est de taille inférieure ou égale à $k \leq k(a+1)$.*

De ces remarques on déduit que toutes les clauses produites ont un corps de taille inférieure ou égale à $\max(k(a+1), s)$.

Montrons que les termes t arguments de prédicats dans les clauses C produites par ces règles à partir de S vérifient (a) ou (b) :

- *Conclusion de la règle de résolution (1) : c'est évident pour les t'_i, en utilisant l'hypothèse appliquée à la prémisse de droite. Pour les t_i, $f(t_1, \ldots, t_n)$ vérifie (a), donc t_i aussi.*
- *Conclusion de la règle de résolution (2') : c'est évident.*
- *Conclusion de la règle de résolution (3) : c'est évident.*
- *Conclusion de la règle de splitting sans splitting : évident.*

Combine a-t-on de clauses dérivées par ces règles ? Par la condition **(b)***, les littéraux de splitting seront toujours de la forme $\ulcorner \perp \Leftarrow B'(Y) \urcorner$, où B' contient au plus k prédicats. Il y a donc au maximum C_p^k littéraux de splitting. A la louche, ceci fait moins de p^k littéraux de splitting, ce qui est encore polynomial, k étant fixé.*

On a le choix parmi $1 + p + pg + p^k$ têtes pour chaque clause ($\perp$, $Q(X)$, $P(f(X_1, \ldots, X_n))$ ou q). Dans le corps de chaque clause, on a au plus $\max(k(a+1), s)$ atomes ; pour chacun, soit c'est un terme sous-terme de S (N possibilités), soit c'est une variable qui n'est pas sous-terme de S (donc sous-terme de la tête, sinon le splitting s'appliquerait, ce qui laisse au plus a possibilités ; ou bien c'est une second conclusion de la règle de splitting sans splitting, ce qui laisse au plus une possibilité). En supposant sans perte de généralité $a \geq 1$, on a donc au plus $(1 + p + pg + p^k)(N + a)^{\max(k(a+1),s)}$ clauses possibles. Le nombre de clauses est donc polynomial, avec un exposant de la forme $O(k + \max(k(a+1), s))$.

7. En déduire que les problèmes de secret, d'accessibilité ($\vdash P$) et généralement tout problème de la forme $\exists X_1, \ldots, X_k \cdot F$ (cf. question 3) dans le spi-calcul se décide, en fait, en temps polynomial.

   Tester l'accessibilité $\vdash P$ revient à tester l'insatisfiabilité de l'ensemble de clauses S calculé en question 1, avec la clause $\perp \Leftarrow \mathrm{acc}_P(*)$. De même, vérifier que l'expression e est secrète revient à vérifier qu'on ne peut pas dériver à la fois $\vdash e = t$, $\vdash c_{pub} = u$, et $u \triangleright t$ pour aucuns termes t et u, autrement dit que S plus la clause $\perp \Leftarrow \mathrm{eq}_e(X), \mathrm{eq}_{c_{pub}}(Y), \mathrm{val}\langle u, t\rangle$ est satisfiable. De même les formules $\exists X_1, \ldots, X_k \cdot F$ de la forme de la question 3 se traduisent en une unique clause. Par la question précédente, et à condition que $k \geq 2$ dans le cas du secret, et k

supérieur ou égal au nombre maximal d'occurrences d'un X_i dans F, ces propriétés sont décidables en temps polynomial. En effet, on ne génère qu'un nombre polynomial de clauses, et l'opération testant si une clause donnée est déjà dans l'ensemble courant de clauses s'effectue en temps polynomial lui aussi. A noter qu'on n'a pas besoin de test de subsomption complet qui, lui, est en général NP-complet.

Deuxième partie (à faire en classe ; durée estimée : 30 min.)

8. On suppose que S est un ensemble satisfiable de clauses de $\mathcal{H}_2^k$. Montrer que l'on peut calculer un modèle de S sous forme d'un automate *non déterministe* (et non alternant). On rappelle qu'un automate non déterministe est un ensemble de clauses de la forme

$$P(f(X_1, \ldots, X_n)) \Leftarrow B_1(X_1), \ldots, B_n(X_n)$$

où les blocs $B_1, \ldots, B_n$ contiennent au plus un prédicat.

> *C'est évident. Par les techniques du cours, un ensemble saturé définit un modèle qui est exactement celui défini par le sous-ensemble des clauses où rien n'est sélectionné, c'est-à-dire le sous-ensemble des clauses d'automate d'arbre alternant. Par la condition* **(a)** *des clauses de $\mathcal{H}_2^k$, ce sont des clauses d'automate d'arbre non déterministe.*

9. On souhaite maintenant vérifier des propriétés d'*authentification*. La propriété d'authentification considérée est la suivante. Soit $e_1\langle e_2\rangle.P$ un sous-processus fixé de A, et Q un sous-processus donné de B. L'authentification de A par B demande à ce que, si l'on atteint Q (si $\vdash Q$ est dérivable), alors A a nécessairement exécuté l'envoi de message e_2 sur le canal e_1 et est passé à P *avant*.

Pour ceci, on se donne un jugement supplémentaire $\vdash P < Q$, pour tout P comme ci-dessus, et tout $Q \in \mathrm{sp}(P_0)$, qui exprime qu'il est possible d'avoir atteint le sous-processus P avant le sous-processus Q. Les règles sont les suivantes :

$$\frac{\vdash e_1\langle e_2\rangle.P \quad \vdash e_1 = t_1 \quad \vdash e_2 = t_2 \quad \vdash e_3(x).Q \quad \vdash e_3 = t_1}{\vdash P < Q}\,(auth)$$

Ensuite les règles :

$$\frac{\vdash P <!Q}{\vdash P < Q}\,(! <) \qquad \frac{\vdash P < (Q_1|Q_2)}{\vdash P < Q_1}\,(| <_1) \qquad \frac{\vdash P < (Q_1|Q_2)}{\vdash P < Q_2}\,(| <_2)$$

$$\frac{\vdash P < (\nu x)Q}{\vdash P < Q}\,(\nu <) \qquad \frac{\begin{array}{c}\vdash P < (\text{case } e_1 \text{ of } \{x\}_{e_2} \Rightarrow Q)\\ \vdash e_1 = \{t\}_{e_2[x_1:=t_1,\ldots,x_k:=t_k]}\\ \vdash x_1 = t_1 \ldots \vdash x_k = t_k\end{array}}{\vdash P < Q}\,(dec <) \qquad \frac{\begin{array}{c}\vdash P < (\text{case } e \text{ of}\\ \langle x, y\rangle \Rightarrow Q)\\ \vdash e = \langle t, u\rangle\end{array}}{\vdash P < Q}\,(proj <)$$

$$\frac{\begin{array}{c}\vdash P < ([e_1 = e_2]Q)\\ \vdash e_1 = t\\ \vdash e_2 = t\end{array}}{\vdash P < Q}\,(=<) \qquad \frac{\vdash P < (e_1\langle e_2\rangle.Q)}{\vdash P < Q}\,(send <) \qquad \frac{\begin{array}{c}\vdash P < (e_1(x).Q)\\ \vdash e_1 = t_1\\ t_1 \triangleright t\end{array}}{\vdash P < Q}\,(recv <)$$

Noter que $<$ n'est pas spécialement définie comme étant transitive.

Etendre le codage de la question 1 pour traiter de ces questions, et montrer que les clauses obtenues sont encore dans $\mathcal{H}_2^k$, pour un certain k. On introduira une constante a_P pour chaque P comme ci-dessus, et on utilisera un prédicat avant_Q tel que $\mathrm{avant}_Q(t)$ est dérivable si et seulement si t est de la forme a_P pour un certain P tel que $\vdash P < Q$ est dérivable par les règles ci-dessus.

Le codage est immédiat :
– $(auth)$ $\mathrm{avant}_Q(a_P) \Leftarrow \mathrm{acc}_P(e_1\langle e_2\rangle.P)(*), \mathrm{eq}_{e_1}(X), \mathrm{eq}_{e_2}(Y), \mathrm{acc}_{e_3(x).Q}(*), \mathrm{eq}_{e_3}(X)$ *;*
– $(! <)$ $\mathrm{avant}_Q(X) \Leftarrow \mathrm{avant}_{!Q}(X)$ *;*
– $(| <_1)$ $\mathrm{avant}_{Q_1}(X) \Leftarrow \mathrm{avant}_{Q_1|Q_2}(X)$ *;*
– $(| <_2)$ $\mathrm{avant}_{Q_2}(X) \Leftarrow \mathrm{avant}_{Q_1|Q_2}(X)$ *;*
– $(\nu <)$ $\mathrm{avant}_Q(X) \Leftarrow \mathrm{avant}_{(\nu x)Q}(X)$ *;*
– $(dec <)$ $\mathrm{avant}_Q(X) \Leftarrow \mathrm{avant}_{\text{case } e_1 \text{ of } \{x\}_{e_2} \Rightarrow Q}(X), \mathrm{eq}_{e_1}(\{Y\}_{e_2[x_1:=X_1,\ldots,x_k:=X_k]}), \mathrm{eq}_{x_1}(X_1), \ldots, \mathrm{eq}_{x_k}$
– $(proj <)$ $\mathrm{avant}_Q(X) \Leftarrow \mathrm{avant}_{\text{case } e_1 \text{ of } \langle x,y\rangle \Rightarrow Q}(X), \mathrm{eq}_e\langle Y, Z\rangle$.
– $(=<)$ $\mathrm{avant}_Q(X) \Leftarrow \mathrm{avant}_{[e_1=e_2]Q}(X), \mathrm{eq}_{e_1}(Y), \mathrm{eq}_{e_2}(Y)$.
– $(send <)$ $\mathrm{avant}_Q(X) \Leftarrow \mathrm{avant}_{e_1\langle e_2\rangle.Q}(X)$ *;*
– $(recv <)$ $\mathrm{avant}_Q(X) \Leftarrow \mathrm{avant}_{e_1(x).Q}(X), \mathrm{eq}_{e_1}(Y), \mathrm{val}\langle Y, Z\rangle$.
Les nouvelles clauses sont dans $\mathcal{H}_2^2$, sauf $(dec <)$ qui est dans $\mathcal{H}_2^{\max(k,1)}$ avec k le nombre maximal d'occurrences des variables x_i dans e_2.

10. Étant donné un ensemble de clauses S, on note $P < Q$ si et seulement si $\mathrm{avant}_Q(a_P)$ est dérivable à partir de S. (Le codage de la question précédente assure que ceci est équivalent à la dérivabilité de $\vdash P < Q$, ceci justifiant la notation.)

    On définit la relation de *causalité* $P \ll Q$ ("P est une cause de Q") comme la relation définie par : $P \ll Q$ si et seulement si l'unique $P' \in \mathrm{sp}(P_0)$ tel que $P' < Q$ est P. Autrement dit, si et seulement si les deux conditions suivantes sont vérifiées :
    – $P < Q$;
    – pour tout $P' \in \mathrm{sp}(P_0)$, si $P' < Q$ alors $P' = P$.
    Montrer que si S est un ensemble de clauses de $\mathcal{H}_2^k$, alors on peut décider $P \ll Q$ en temps polynomial en la taille de S et le cardinal $|\mathrm{sp}(P_0)|$ de $\mathrm{sp}(P_0)$, à condition de supposer fixés certains paramètres que l'on précisera.

    Tester $P < Q$ s'effectue en temps polynomial, par test de satisfiabilité d'ensembles de clauses de $\mathcal{H}_2^k$. Il ne reste plus qu'à vérifier que $P' < Q$ est faux pour tout $P' \in \mathrm{sp}(P_0) \setminus \{P\}$, chaque test prenant un temps polynomial. Ce temps polynomial est, comme plus haut, à k, s, et a fixés ; ici s est le maximum de la taille maximale des corps de clauses de S et de la taille maximale des clauses de la question 9, qui est à une constante additive près borné par deux fois la taille de la plus grande expression e_2 utilisée comme clé de déchiffrement. Le nombre de tels tests est $|\mathrm{sp}(P_0)|$. D'où la conclusion.

    On peut aussi optimiser en compilant d'abord S sous forme d'un automate non déterministe comme en question 8.

11. En déduire, ainsi que des questions précédentes, que la question de l'authentification est décidable en temps polynomial pour les processus du spi-calcul, un certain paramètre étant fixé.

    Étant donnés deux sous-processus $e_1\langle e_2\rangle.P$ de A et Q de B, l'authentification de A par B est équivalente à $P \ll Q$, et on applique la question précédente. Le paramètre à fixer est la taille maximale des clés de déchiffrement.