Calculabilité. Devoir à rendre au plus tard le 16 octobre 2017

Contrôles d'accès et politiques de sécurité

L'objet du problème est d'étudier des systèmes de contrôle d'accès, typiquement ceux qui sont utilisés dans les systèmes d'exploitation. Dans la suite U désignera un ensemble (infini) d'*utilisateurs*, $O \supseteq U$ un ensemble d'*objets* et R un ensemble fini de *droits*.

Dans la suite, si σ est une application d'une séquence de variables $\vec{x}$ dans un ensemble E et e est une expression contenant des variables de $\vec{x}$, $e\sigma$ désigne l'expression e dans laquelle chaque variable $x \in \vec{x}$ est substituée par $\sigma(x)$.

Un système de contrôle d'accès (SCA) est défini par un ensemble fini d'utilisateurs $u \subseteq U$, un ensemble fini d'objets $o \subseteq O$, contenant u, et une matrice D de droits d'accès, qui, à chaque utilisateur et chaque objet associe un sous-ensemble de R.

Un ensemble fini de *commandes* permettent de modifier la matrice des droits d'accès. C'est la *politique de sécurité*. De manière générale, une commande c est paramétrée par un nombre fini de variables $\vec{x} = x_1, \ldots, x_n$ de type O et est définie par:

- $\mathsf{PRECOND}_c(\vec{x}) \subseteq \vec{x} \times \vec{x} \times R$
- $\mathsf{NEWU}_c(\vec{x}) \subseteq \vec{x}$
- $\mathsf{NEWO}_c(\vec{x}) \subseteq \vec{x}$
- $\mathsf{DELO}_c \subseteq \vec{x}$
- $\mathsf{DELU}_c \subseteq \vec{x}$
- $\mathsf{GRANT}_c(\vec{x}) \subseteq \vec{x} \times \vec{x} \times R$
- $\mathsf{REM}_c(\vec{x}) \subseteq \vec{x} \times \vec{x} \times R$

Les commandes agissent sur les SCA de la manière suivante. Si $(u, o, D$ est un SCA, c une commande paramétrée par $\vec{x}$ et σ est une application de $\vec{x}$ dans O, l'effet de la commande $c\sigma$ sur (u, o, D) est un SCA (u', o', D') défini comme suit:

1. Si $\exists (a, b, r) \in \mathsf{PRECOND}_c(\vec{x})\sigma$, $(a, b) \notin u \times o$ ou $r \notin D(a, b)$, alors la commande est sans effet.
2. Si $\mathsf{NEWU}_c(\vec{x})\sigma \cap u \neq \emptyset$ ou $\mathsf{NEWO}_c(\vec{x})\sigma \cap o \neq \emptyset$ ou $\mathsf{DELU}_c(\vec{x})\sigma \not\subseteq u$ ou $\mathsf{DELO}_c(\vec{x})\sigma \not\subseteq o$, la commande est sans effet.

   Sinon, soient $u' = (u \cup \mathsf{NEWU}_c(\vec{x})\sigma) \setminus \mathsf{DELU}_c(\vec{x})\sigma$, $o' = (o \cup \mathsf{NEWO}_c(\vec{x})\sigma \cup \mathsf{NEWU}_c(\vec{x})\sigma) \setminus \mathsf{DELO}_c(\vec{x})\sigma$. Soit D' défini par $D'(a, b) = D(a, b)$ si $(a, b) \in (u \cap u') \times (o \cap o')$ et $D'(a, b) = \emptyset$ si $(a, b) \in (u' \times o') \setminus (u \times o)$.
3. $\forall (a, b, r) \in \mathsf{REM}_c(\vec{x})\sigma$, si $(a, b) \notin u' \times o'$, la commande est sans effet. Sinon, r est retiré de $D'(a, b)$
4. $\forall (a, b, r) \in \mathsf{GRANT}_c(\vec{x})\sigma$, si $(a, b) \notin u' \times o'$, la commande est sans effet. Sinon, r est ajouté à $D'(a, b)$.

Question 0

Soient $u = o = \{\mathsf{root}, \mathsf{alice}, \mathsf{charlie}\}$ et $D(x, x) = R = \{\mathsf{o}, \mathsf{c}, \mathsf{r}, \mathsf{w}, \mathsf{d}\} = D(\mathsf{root}, x)$ pour tout $x \in u$.

La politique de sécurité est donnée par les commandes suivantes:

	PRECOND	NEWU	NEWO	DELU	DELO	GRANT	REM
$\mathsf{newu}_S(x, y, z)$	(x, y, c)	z	z	$-$	$-$	$(y, z, s), s \in S$	$-$
	$(x, y, s), s \in S$	$-$	$-$	$-$	$-$		$-$
$\mathsf{chmod}_{+s}(x, y)$	(x, y, o)	$-$	$-$	$-$	$-$	(x, y, s)	$-$
$\mathsf{chmod}_{-s}(x, y)$	(x, y, o)	$-$	$-$	$-$	$-$	$-$	(x, y, s)
$\mathsf{remu}(x, y)$	(x, y, d)	$-$	$-$	y	y	$-$	$-$

Montrer que les utilisateurs alice et charlie peuvent avoir tous deux les mêmes droits sur certains objets.

Question 1

Montrer que le problème suivant est indécidable:

Donnée: un SCA (u, o, D), un ensemble fini de droits R, une politique de sécurité, un utilisateur $a \in u$, un objet $b \in o$, une permission $r \in R$.

Question: Existe-t-il une suite d'instances de commandes qui conduit à un SCA dans lequel a possède la permission r pour l'objet b ?

Question 2

Dans cette question, l'ensemble de droits R est fixé.

Montrer qu'il existe un ensemble de droits R pour lequel le problème suivant est indécidable:

Donnée: un SCA (u, o, D), une politique de sécurité, un utilisateur $a \in u$, un objet $b \in o$, une permission $r \in R$.

Question: Existe-t-il une suite d'instances de commandes qui conduit à un SCA dans lequel a possède la permission r pour l'objet b ?

Question 3

Dans cette question, l'ensemble de droits R et la politique de sécurité sont fixés.

Montrer qu'il existe un ensemble de droits R et une politique de sécurité pour lesquels le problème suivant est indécidable:

Donnée: un SCA (u, o, D), un utilisateur $a \in u$, un objet $b \in o$, une permission $r \in R$.

Question: Existe-t-il une suite d'instances de commandes qui conduit à un SCA dans lequel a possède la permission r pour l'objet b ?

Question 4

On s'interdit ici les commandes pour lesquelles $\mathsf{NEWU} \neq \emptyset$ (autrement dit, on ne peut pas créer d'utilisateur). En revanche, on ajoute dans la définition d'une commande c un ensemble $\mathsf{UNSET}_c \subseteq \vec{x}$, avec la sémantique suivante:

> Si $\exists(a, b, r) \in \mathsf{UNSET}_c(\vec{x})\sigma$ et $r \in D(a, b)$, alors la commande est sans effet

Autrement dit les préconditions ne sont plus seulement l'existence de droits mais aussi l'absence de droits.

Montrer qu'il existe un ensemble de droits tel que le problème suivant est indécidable:

Donnée: une politique de sécurité qui ne permet aucune création d'utilisateur, $a \in U, b \in O, r \in R$

Question: pour tout SCA (u, o, D) tel que $a \in u, b \in o$, il existe une suite de commandes qui conduit à octroyer la permission r à a pour l'objet b.

Solution

Question 1

Commentaire sur la question: L'idée générale est de représenter chaque case du ruban par un utilisateur. Les droits d'accès donnent les états ou symboles du ruban. La difficulté ici vient du fait que l'ensemble des utilisateurs n'est pas ordonné: il n'y a pas de "utilisateur suivant". Il fallait donc coder l'ordre des cases du ruban à l'aide de droits spécifiques. Environ 70% des élèves n'ont pas vu cette difficulté et donc proposé un codage incorrect.

On réduit le problème suivant:

Donnée: une machine de Turing M qui n'écrit jamais de blancs et possède un unique état final q_f et ne fait aucun mouvement sur place

Question: M s'arrête sur la donnée vide, avec la tête de lecture en début de ruban

Soit $M = (Q, \Sigma, q_0, \$, B, \delta)$. On choisit $U = O = \mathbb{N}$ (mais on peut choisir n'importe quel ensemble infini; dans toute la suite, on peut renommer les utilisateurs), $R = \{\mathsf{p}, \mathsf{n}\} \cup Q \cup \Sigma$, $u = \{0, 1\} = o$, $D_0(0,0) = \{q_0, \$\}$, $D_0(0,1) = \{\mathsf{n}\}$, $D_0(1,0) = \{\mathsf{p}\}$, $D_0(1,1) = \{B\}$, $a = b = 0$, $r = q_f$.

Pour chaque transition $\delta(q, a) = (q', a', d)$, on construit une commande $c_{q,a,q',a',d}$:

	PRECOND	NEWU	NEWO	DELU	DELO	GRANT	REM
$c_{q,a,q',a',\rightarrow}$	(x,x,q)	$-$	$-$	$-$	$-$	(x,x,a')	(x,x,q)
$a \neq B$	(x,x,a)	$-$	$-$	$-$	$-$	(y,y,q')	(x,x,a)
	(x,y,n)	$-$	$-$	$-$	$-$	$-$	$-$
$c_{q,B,q',a',\rightarrow}$	(x,x,q)	y	y	$-$	$-$	(y,y,q')	(x,x,q)
	(x,x,B)	$-$	$-$	$-$	$-$	(x,y,n)	(x,x,B)
	$-$	$-$	$-$	$-$	$-$	(y,x,p)	$-$
	$-$	$-$	$-$	$-$	$-$	(y,y,B)	$-$
	$-$	$-$	$-$	$-$	$-$	(x,x,a')	$-$
$c_{q,a,q',a',\leftarrow}$	(x,x,q)	$-$	$-$	$-$	$-$	(y,y,q')	(x,x,q)
$a \neq B$	(x,y,p)	$-$	$-$	$-$	$-$	(x,x,a')	(x,x,a)
	(x,x,a)	$-$	$-$	$-$	$-$	$-$	$-$
$c_{q,B,q',a',\leftarrow}$	(x,x,q)	y	y	$-$	$-$	(z,z,q')	(x,x,q)
	(x,x,B)	$-$	$-$	$-$	$-$	(x,y,n)	(x,x,B)
	(x,z,p)	$-$	$-$	$-$	$-$	(y,x,p)	$-$
	$-$	$-$	$-$	$-$	$-$	(y,y,B)	$-$
	$-$	$-$	$-$	$-$	$-$	(x,x,a')	$-$

NB: dans cette définition, le cas $a = a'$ est traité correctement gâce à la sémantique: on commence par retirer des droits avant d'en ajouter.

La construction de cette politique de sécurité est récursive.

Si $\gamma = a_0 a_1 \cdots a_{k-1} q a_k \cdots a_{k+m}$ est une configuration de la machine de Turing (avec $a_0 = \$$, $a_{k+m} = B$ et $a_1, \ldots, a_{k+m-1} \neq B$), on note $s(\gamma)$ le SCA dans lequel $u = o = \{0, \ldots, k+m\}$ et

- $D(i,i) = \{a_i\}$ pour tout $i \neq k+m$,

- $D(i, i+1) = \{\mathsf{n}\}$ pour $0 \leq i \leq k+m-1$
- $D(i, i-1) = \{\mathsf{p}\}$, pour $1 \leq i \leq k+m$
- $D(k, k) = \{q, a_k\}$
- $D(i, j) = \emptyset$ dans tous les autres cas

On montre par récurrence sur n que:

- Si $q_0\$B \vdash^n \gamma_n$ alors il existe une séquence (d'instances) de n commandes $c_1, \ldots, c_n$ telles que $c_n(\cdots(c_1(D_0))\cdots) = s(\gamma_n)$
- Réciproquement, si $c_1, \ldots, c_n$ est une suite (d'instances) de commandes telles que $D_n = c_n(\cdots(c_1(D_0))\cdots)$, alors $D_n = s(\gamma_n)$ et $q_0\$B \vdash^n \gamma_n$.

Pour $n = 0$, il suffit de vérifier que $s(q_0\$B) = D_0$.

Pour l'étape de récurrence, on vérifie qu'une commande au plus est applicable à $s(\gamma)$: dans tous les cas $(x, x, q) \in \mathsf{PRECOND}$ or, au plus un utilisateur k possède le droit q. La seule instance possible est donc $x\sigma = k$. Par ailleurs, pour tout utilisateur i, il existe au plus un j tel que $D_n(i, j) = \mathsf{n}$ et, dans ce cas, $j = i + 1$ (resp. $D_n(i, j) = \mathsf{p}$ et, dans ce cas, $j = i - 1$).

- Si $m \neq 0$ et $D_n(k, k) = \{q, a\}$, la seule commande applicable est $c_{q,a,\delta(q,a)}$. Le SCA D_{n+1} résultant de l'application de cette commande vérifie par construction $D_{n+1}(i, i) = D_n(i, i)$ pour $i \notin \{k, k+1\}$ si $\delta(q, a) = q', a', \rightarrow$ (resp. pour $i \notin \{k, k-1\}$ si $\delta(q, a) = q', a', \leftarrow$) et $D_{n+1}(k, k) = \{a'\}$, $D_{n+1}(k+1, k+1) = D_n(k+1, k+1) \cup \{q'\}$ (resp. $D_{n+1}(k-1, k-1) = D_n(k-1, k-1) \cup \{q'\}$). Dans tous les cas, $D_{n+1} = s(\gamma')$ où $\gamma \vdash_M \gamma'$.
- Si $m = 0$ (tête de lecture en fin de ruban), D_{n+1} contient un nouvel utilisateur, que nous noterons $k + 1$ sans perte de généralité (mais attention: il n'a a priori aucun lien avec l'utilisateur k). Tout se passe comme si, dans le cas précédent, on avait $D_n(k+1, k+1) = B$.

Si M s'arrête sur le mot vide: $\gamma_0 \vdash_M^n \gamma_n$ où $\gamma_n = q_f\$w_n$, alors il existe une suite (d'instances) de commandes $c_1, \ldots, c_n$ telle que $c_n(\cdots(c_1(s(\gamma_0)))\cdots) = s(\gamma_n)$. Or $q_f \in s(\gamma_n)(0, 0)$: a possède la permission r pour b.

Réciproquement, s'il existe une suite (d'instances) de commandes $c_1, \ldots, c_n$ telles que $D_n = c_n(\cdots(c_1(D_0))\cdots)$ et $q_f \in D_n(0, 0)$, alors $D_n = s(\gamma_n)$ avec $\gamma_0 \vdash_M^n \gamma_n$ et γ_n est une configuration d'arrêt.

Question 2

On réduit le problème de l'arrêt sur le mot vide. On se donne ainsi une machine de Turing M. On suppose sans perte de généralité que M n'écrit jamais B ou \$et que M possède un unique état d'arrêt q_a, qui ne peut être atteint qu'en début de ruban : il n'existe qu'une seule transition qui mène à q_a et c'est une transition $q, \$ \rightarrow q_a, \$, \downarrow$.

L'idée est ici que les états, les symboles du ruban et les transitions soient codés par des objets/utilisateurs supplémentaires.

On utilise $R = \{\mathsf{rB}, \mathsf{t}^{\rightarrow}_{Q}, \mathsf{t}^{\rightarrow}_{nQ}, \mathsf{t}^{\rightarrow}_{S}, \mathsf{t}^{\rightarrow}_{nS}, \mathsf{t}^{\leftarrow}_{Q}, \mathsf{t}^{\leftarrow}_{S}, \mathsf{t}^{\leftarrow}_{nQ}, \mathsf{t}^{\leftarrow}_{nS}, \mathsf{t}^{\downarrow}_{Q}, \mathsf{t}^{\downarrow}_{S}, \mathsf{t}^{\downarrow}_{nQ}, \mathsf{t}^{\downarrow}_{nS}, \mathsf{cq}, \mathsf{cs}, \mathsf{o}, \mathsf{n}, \mathsf{p}\}$. (On pourrait utiliser un ensemble de droits beaucoup plus réduit (4 droits suffisent), mais on ne cherche pas ici à optimiser).

Les utilisateurs sont:

- Le symbole blanc B
- les transitions de la machine
- les cases du ruban

Les objets, outre les utilisateurs ci-dessus, contiennent les symboles et les états.

Une case du ruban $i \in \mathbb{N}$, possède un droit cq sur l'objet x si et seulement si $x = q \in Q$, la machine est dans l'état q et la tête de lecture en i.

Une case du ruban $i \in \mathbb{N}$, possède un droit cs sur l'objet x si et seulement si $x = a \in \Sigma$, et la $i$ème case du ruban contien le symbole a.

Pour chaque transition $\delta(q, a) = (q', a', d)$, l'utilisateur correspondant possède le droit $\mathsf{t}^{\rightarrow}_{Q}$ (ou $\mathsf{t}^{\leftarrow}_{Q}$ ou $\mathsf{t}^{\downarrow}_{Q}$ suivant d) sur q, le droit $\mathsf{t}^{\rightarrow}_{S}$ (ou $\mathsf{t}^{\leftarrow}_{S}$ ou $\mathsf{t}^{\downarrow}_{S}$) sur a, le droit $\mathsf{t}^{\rightarrow}_{nQ}$ sur q' (resp. $\mathsf{t}^{\leftarrow}_{nQ}$, resp. $\mathsf{t}^{\downarrow}_{nQ}$) , le droit $\mathsf{t}^{\rightarrow}_{nS}$ (resp. $\mathsf{t}^{\leftarrow}_{nS}$, resp. $\mathsf{t}^{\downarrow}_{nS}$) sur a'.

Si $\gamma = a_0 \cdots a_{k-1} q a_k \cdots a_{k+m}$, $s(\gamma)$ est donné par

	q_0 $\cdots$ q $\cdots$ q' $\cdots$ q_a	\$ $\cdots$ a $\cdots$ a' $\cdots$ B	$\cdots$ $\delta(q,a)$ $\cdots$	$\cdots$ $k-1$ k $k+1$ $\cdots$
B		rB		
$\vdots$ $\delta(q,a)$ $\vdots$	$\mathsf{t}^{\rightarrow}_{Q}$ $\mathsf{t}^{\rightarrow}_{nQ}$	$\mathsf{t}^{\rightarrow}_{S}$ $\mathsf{t}^{\rightarrow}_{nS}$		
$\vdots$ $k-1$ k $k+1$ $\vdots$	cq	cs		o n p o n p o

Un utilisateur k exactement a un droit cq, et c'est un droit sur un symbole détat q: k est la position de la tête de lecture et q l'état de la machine. Le symbole a pointé par la tête de lecture est le seul sur lequel k a le droit cs.

La politique de sécurité comprend une commande par règle de transition. Par exemple, si $\delta(q, a) = (q', a', \rightarrow)$, $(x, y, \mathsf{cq}), (x, z, \mathsf{cs}) \in \mathsf{PRECOND}$, avec dans l'idée que x doit être une case du ruban, y un état et z un symbole de ruban. $(x, x, \mathsf{o}) \in \mathsf{PRECOND}$ force l'instance de x à être une case du ruban. $(x, y, \mathsf{cq}) \in \mathsf{PRECOND}$ force x à être la position de la tête de lecture et l'instance de y à être un état. $(x, z, \mathsf{cs}) \in \mathsf{PRECOND}$ force l'instance de z à être le symbole de ruban sous la tête de lecture. $(t, y, \mathsf{t}^{\rightarrow}_{Q}) \in \mathsf{PRECOND}$ et $(t, z, \mathsf{t}^{\rightarrow}_{S}) \in \mathsf{PRECOND}$ forcent t à être la transition $\delta(q, a)$ applicable dans la configuration codée dans le SCA. $(t, y', \mathsf{t}^{\rightarrow}_{nQ}) \in \mathsf{PRECOND}$ force $y' = q'$ et $(t, z', \mathsf{t}^{\rightarrow}_{nS}) \in \mathsf{PRECOND}$ force $z' = a'$. Le chainage entre cases du ruban est assuré, comme dans la question précédente, par les droits p, n.

Pour chaque transition $\delta(q,a) = (q', a', d)$, on construit une commande:

	PRECOND	NEWU	NEWO	DELU	DELO	GRANT	REM
$c_{q,a,q',a',\rightarrow}$	(x, y, cq)	–	–	–	–	(x', y', cq)	(x, y, cq)
$a \neq B$	(x, z, cs)	–	–	–	–	(x, z', cs)	(x, z, cs)
	$(t, z, \mathsf{t}_S^{\rightarrow})$	–	–	–	–	–	–
	$(t, y, \mathsf{t}_Q^{\rightarrow})$	–	–	–	–	–	–
	$(t, y', \mathsf{t}_{nQ}^{\rightarrow})$	–	–	–	–	–	–
	$(t, z', \mathsf{t}_{nS}^{\rightarrow})$	–	–	–	–	–	–
	(x, x', n)	–	–	–	–	–	–
$c_{q,B,q',a',\rightarrow}$	(x, y, cq)	x'	x'	–	–	(x', y', cq)	(x, y, cq)
	(x, z, cs)	–	–	–	–	(x, z', cs)	(x, z, cs)
	(z, z, rB)	–	–	–	–	(x', z, cs)	–
	$(t, z, \mathsf{t}_S^{\rightarrow})$	–	–	–	–	(x, x', n)	–
	$(t, y, \mathsf{t}_Q^{\rightarrow})$	–	–	–	–	(x', x', o)	–
	$(t, y', \mathsf{t}_{nQ}^{\rightarrow})$	–	–	–	–	(x', x, p)	–
	$(t, z', \mathsf{t}_{nS}^{\rightarrow})$	–	–	–	–	–	–
$c_{q,a,q',a',\leftarrow}$	(x, y, cq)	–	–	–	–	(x', y', cq)	(x, y, cq)
$a \neq B$	(x, z, cs)	–	–	–	–	(x, z', cs)	(x, z, cs)
	$(t, y, \mathsf{t}_Q^{\leftarrow})$	–	–	–	–	–	–
	$(t, z, \mathsf{t}_S^{\leftarrow})$	–	–	–	–	–	–
	$(t, y', \mathsf{t}_{nQ}^{\leftarrow})$	–	–	–	–	–	–
	$(t, z', \mathsf{t}_{nS}^{\leftarrow})$	–	–	–	–	–	–
	(x, x', p)	–	–	–	–	–	–
$c_{q,B,q',a',\leftarrow}$	(x, y, cq)	x'	x'	–	–	(x'', y', cq)	(x, y, cq)
	(x, z, cs)	–	–	–	–	(x, z', cs)	(x, z, cs)
	(z, z, rB)	–	–	–	–	(x', z, cs)	–
	$(t, z, \mathsf{t}_S^{\leftarrow})$	–	–	–	–	(x, x', n)	–
	$(t, y, \mathsf{t}_Q^{\leftarrow})$	–	–	–	–	(x', x', o)	–
	$(t, y', \mathsf{t}_{nQ}^{\leftarrow})$	–	–	–	–	(x', x, p)	–
	$(t, z', \mathsf{t}_{nS}^{\leftarrow})$	–	–	–	–	–	–
	(x, x'', p)	–	–	–	–	–	–
$c_{q,a,q',a',\downarrow}$	(x, y, cq)	–	–	–	–	(x, y', cq)	(x, y, cq)
$a \neq B$	(x, z, cs)	–	–	–	–	(x, z', cs)	(x, z, cs)
	$(t, y, \mathsf{t}_Q^{\downarrow})$	–	–	–	–	–	–
	$(t, z, \mathsf{t}_S^{\downarrow})$	–	–	–	–	–	–
	$(t, y', \mathsf{t}_{nQ}^{\downarrow})$	–	–	–	–	–	–
	$(t, z', \mathsf{t}_{nS}^{\downarrow})$	–	–	–	–	–	–
$c_{q,B,q',a',\downarrow}$	(x, y, cq)	x'	x'	–	–	(x, y', cq)	(x, y, cq)
	(x, z, cs)	–	–	–	–	(x, z', cs)	(x, z, cs)
	$(t, y, \mathsf{t}_Q^{\downarrow})$	–	–	–	–	(x, x', n)	–
	$(t, z, \mathsf{t}_S^{\downarrow})$	–	–	–	–	(x', x', o)	–
	$(t, y', \mathsf{t}_{nQ}^{\downarrow})$	–	–	–	–	(x', x, p)	–
	$(t, z', \mathsf{t}_{nS}^{\downarrow})$	–	–	–	–	(x', z, cs)	–
	(z, z, rB)	–	–	–	–	–	–

La réduction consiste, à partir de la machine M, à calculer la politique de sécurité comme ci-dessus, ainsi que le SCA correspondant à la configuration initiale: si $M = (Q, \Sigma, q_0, \{\$, B\}, \delta)$, $u = \{B\} \uplus Q \times \Sigma \uplus \{0, 1\}$, $o = u \uplus \Sigma \setminus \{B\} \uplus Q$ et

- $D_0(B, B) = \{\mathsf{rB}\}$ et $D_0(B, x) = \emptyset$ si $x \neq B$
- $D_0((q, a), a) = \{\mathsf{t}_S^{\rightarrow}\}$ (resp. $\{\mathsf{t}_S^{\leftarrow}\}$, resp. $\{\mathsf{t}_S^{\downarrow}\}$) et $D_0((q, a), q) = \{\mathsf{t}_Q^{\rightarrow}\}$ (resp. $\{\mathsf{t}_Q^{\leftarrow}\}$, resp. $\{\mathsf{t}_Q^{\downarrow}\}$) et $D_0((q, a), q') = \{\mathsf{t}_{nQ}^{\rightarrow}\}$ (resp. $\{\mathsf{t}_{nQ}^{\leftarrow}\}$, resp. $\{\mathsf{t}_{nQ}^{\downarrow}\}$) et $D_0((q, a), a') = \{\mathsf{t}_{nS}^{\rightarrow}\}$ (resp. $\{\mathsf{t}_{nS}^{\leftarrow}\}$, resp. $\{\mathsf{t}_{nS}^{\downarrow}\}$) si $\delta(q, a) = (q', a', \rightarrow)$ (resp. $\delta(q, a) = (q', a', \leftarrow)$, resp. $\delta(q, a) = (q', a', \downarrow)$).

  $D_0((q, a), x) = \emptyset$ dans les autres cas.
- $D_0(0, 0) = \{\mathsf{o}\}, D_0(0, 1) = \{\mathsf{n}\}$, $D_0(0, \$) = \{\mathsf{cs}\}$, $D_0(0, q_0) = \{\mathsf{cq}\}$, $D_0(0, x) = \emptyset$ dans les autres cas.
- $D_0(1, 0) = \{\mathsf{p}\}, D_0(1, 1) = \{\mathsf{o}\}$, $D_0(1, B) = \{\mathsf{cs}\}$, $D_0(1, x) = \emptyset$ dans les autres cas.

Finalement, on choisit pour objet b l'état d'arrêt q_a, pour utilisateur a, la case initiale 0 et pour droit r cq.

Question 3

En fait, la construction de la question 2 fournit la réponse, car la politique de sécurité construite ne dépend pas de la machine de Turing: elle contient toujours 6 commandes distinctes.

Question 4

Le problème n'est pas résolu (solution fausse).

L'idée générale était de réduire l'arrêt universel des automates linéairement bornés: à une telle machine M on peut associer une politique de sécurité qui permet de simuler les mouvements de la machine, dans l'esprit de la question 2 (mais en utilisant une politique de sécurité qui dépend de la machine M). Un automate linéairement borné, dans n'importe quel état q, en lisant B, ou bien s'arrête, ou bien déplace la tête de lecture vers la gauche. Par conséquent, on peut supprimer, dans la réduction, les commandes $c_{q,B,q',a',\rightarrow}$ sans changer la correction de la réduction. Après avoir supprimé ces commandes, il n'y a plus de création d'utilisateur.

Néanmoins, les SCA ne correspondent pas nécessairement à des configurations d'une machine de Turing. Il se pourrait donc que la réponse à la question de l'énoncé soit "non" parce que, pour certains SCA, il n'y a pas de séquence de commandes qui conduisent à octroyer le droit r à a pour b, mais que ces SCA ne correspondent pas à des configurations initiales d'ALB. Il faut donc ajouter des commandes qui testent qu'un SCA code la configuration initiale d'une MT et, si ce n'est pas le cas, donnent le droit r à a pour b. Pour cela l'idée est d'utiliser UNSET.

On partitionne d'abord les utilisateurs en un nombre fixe de types différents $\mathsf{o}_1, \ldots, \mathsf{o}_m$ et on vérifie que chacun d'eux a les droits sur lui-même pour son type, mais pas pour les autres.

Par exemple, chaque utilisateur possède au moins un type:

$$\mathsf{UNSET}_c = \{(x, x, \mathsf{o}_1), \ldots, (x, x, \mathsf{o}_m)\}, \mathsf{GRANT}_c = \{(y, z, r)\}$$

Chaque utilisateur n'a qu'un seul type: pour $i \neq j$,

$$\mathsf{PRECOND}_{c_{i,j}} = \{(x, x, \mathsf{o}_i), (x, x, \mathsf{o}_j)\}, \mathsf{GRANT}_c = \{(y, z, r)\}$$

On peut aussi vérifier qu'un utilisateur de type "case de ruban" (caractérisé par o_1) a des droits correspondant à au plus un état (caractérisé par o_2) et donc que la tête de lecture est à une position au plus sur le ruban:

	PRECOND	UNSET	GRANT	REM
c_0	(x, x, o_1)	$-$	(x, x, o_1)	(x, x, o_1)
	(y, y, o_1)	$-$	(y, y, o_1)	(y, y, o_1)
	(x, y, o_1)	$-$	$-$	(x, y, o_1)
c_1	(x, x_1, cq)	(x, y, o_1)	(x', y', r)	$-$
	(y, y_1, cq)	$-$	$-$	$-$
	(x, x, o_1)	$-$	$-$	$-$
	(y, y, o_1)	$-$	$-$	$-$
c_2	(x, x_1, cq)	(x, x, o_1)	(x', y', r)	$-$
c_3	(x, x_1, cq)	(x_1, x_1, o_2)	(x', y', r)	$-$

Si plusieurs utilisateurs (distincts) ont les droits cq, alors, en utilisant c_0, on peut se ramener au cas où x n'a pas les droits o_1 sur y, puis utiliser la commande c_1 pour donner le droit r à a sur b. Les commandes c_2, c_3 servent ici uniquement à vérifier les types des variables.

De même, on peut s'assurer que chaque utilisateur (correspondant à une case du ruban) a les droits sur un seul symbole de ruban. On peut aussi vérifier le chainage des cases du ruban.

Il est donc possible de vérifier qu'un SCA code une configuration de M (et donner le droit r à a sur b si ce n'est pas le cas).

En revanche (et c'est là qu'est l'erreur de l'auteur, désolé), on ne peut pas (ou au moins pas de manière simple) vérifier qu'un SCA code une configuration accessible de M. En fait, on ne peut pas tester qu'un SCA code une configuration *initiale* de M. La méthode marcherait si on pouvait montrer que l'arrêt universel des ALB *depuis n'importe quelle configuration* est indécidable.