Université Paul Sabatier, Toulouse III
Master 2 - Recherche
Intelligence artificielle : raisonnement, coopération, langage
Responsable : Claudette Cayrol, professeur
2004-2005

Institut de recherche en informatique de Toulouse - IRIT
Equipe Logique, Interaction, Langue, et Calcul - LILaC

LE PROBLEME DE LA DISPONIBILITE DANS LES SYSTEMES DE PROTECTION TEMPORISEE

par Fahima Cheikh

Directeur de recherche : Philippe Balbiani

Résumé : Ce travail s'inscrit dans le cadre de la sécurité informatique et plus précisément dans le domaine du contrôle d'accès. Différents modèles existent pour formaliser le contrôle d'accès ainsi que le problème de la protection dans les systèmes d'exploitation. Aucun de ces modèles ne prend en considération la notion de la disponibilité qui est l'une des notions les plus importantes dans la sécurité informatique. C'est pourquoi nous proposons dans ce rapport une approche pour la modélisation des systèmes de protection temporisée dans laquelle on peut définir la disponibilité. Cette approche est une extension du modèle matriciel introduit par Harrison, Ruzzo et Hullman en 1976.

Mots-clés : Contrôle d'accès, modèle matriciel, raisonnement temporel, disponibilité.

Abstract : In the research area of computer security, this work explores the access control domain. Different models exist to formalize the access control and the protection problem in operating systems. None of these models consider the availability notion which is one of the most important notions in computer security. Therefore we propose in this report an approach to modelling timed protection systems in which we can define availability. This approach is an extension of the matrix model introduced by Harrison, Ruzzo and Hullman in 1976.

Key-words : Access control, matrix model, temporal reasoning, availability.

Remerciements

Ce travail n'aurait pu voir le jour sans la contribution de nombreuses personnes, dont je me fais aujourd'hui un plaisir et un devoir de les remercier.

J'exprime mon profond remerciement à mon directeur de recherche, Philippe Balbiani, pour toute l'attention qu'il m'a accordée durant ce travail, pour sa disponibilité et surtout pour m'avoir beaucoup appris.

Je tiens à remercier Yannick Chevalier, pour m'avoir aidé à faire les démonstrations de théorèmes et aussi pour sa gentillesse.

Je remercie tous les membres de l'équipe LILaC pour leur disponibilité.

Enfin, que tous ceux qui ont contribué, de près ou de loin, à l'élaboration de ce travail trouvent ici l'expression de ma gratitude.

Table des matières

Notations

Dans ce rapport, les notations suivantes seront utilisées :

Notations concernant tout le rapport
HRU : Harrison, Ruzzo et Ullman
SC : ensemble dénombrable de constantes de type sujet
s : élément de SC
S : sous-ensemble de SC
SV : ensemble dénombrable de variables de type sujet
x : élément de SV
OC : ensemble dénombrable de constantes de type objet
o : élément de OC
O : sous-ensemble de OC
OV : ensemble dénombrable de variables de type objet
y : élément de OV
R : ensemble fini de droit
r : élément de R
π : opération primitive
θ : assignation
Δ : état de protection
A : matrice des actions
B : matrice des buts
P : matrice des permissions
h : histoire
d : durée de temps
v : instant
$\mathcal{C}$: classe des systèmes de protection
$\mathcal{C}(i,j)$: classe des systèmes de protection dont aucune des commandes ne contient plus de i conditions ou plus de j opérations primitives
$\mathcal{C}^+(i,j)$: classe des systèmes de protection monotones appartenant à $\mathcal{C}(i,j)$
Notations concernant le chapitre 2
P : matrice d'accès
Π : système de protection
Notation concernant le chapitre 3
Π : système de protection temporisée
Notation concernant le chapitre 4
Π : système de protection temporisée spécifique

Introduction

La sécurité informatique consiste à protéger les informations contre les malveillances intentionnelles des utilisateurs. Il existe trois concepts importants dans la sécurité informatique : la confidentialité, l'intégrité et la disponibilité. Le concept de confidentialité traite des permissions et des interdictions faites aux utilisateurs de connaître certaines informations. Le concept d'intégrité quant à lui caractérise la légalité des actions qui visent à mettre à jour ou à détruire les informations. Le troisième concept, celui de la disponibilité, aborde la possibilité d'utiliser les ressources désirées dans les délais spécifiés. Les techniques et les théories de l'intelligence artificielle devraient être capables de s'attaquer aux problèmes posés dans le domaine de la sécurité informatique. En effet le concept de la confidentialité est basé sur des notions déontiques [8] et épistémiques [11] abordées par l'intelligence artificielle dans le cadre de la représentation des connaissances et de la formalisation du raisonnement. Dans ce même cadre, il existe la théorie de la mise à jour [15] qui peut être utile pour formaliser les problèmes liés au concept de l'intégrité. Aborder la question de la disponibilité, c'est traiter des questions semblables à celles considérées dans le contexte de la planification [19] ou du raisonnement temporel [2]. Par conséquent l'intelligence artificielle a également développé des modèles de raisonnement qui pourraient être apropriés à la formalisation du raisonnement sur la disponibilité.

Dans un système d'exploitation comme Unix ou Windows, les utilisateurs du système ont des droits d'accès comme $read$ et $write$ sur des fichiers. L'état du système d'exploitation peut évoluer grâce à des commandes. Par exemple, dans le système d'exploitation Unix, pour détruire un fichier on utilise la commande rm, pour le créer on utilise la commande $touch$ et pour changer les droits d'accès en utilise la commande $chmod$. Les informations contenues dans les fichiers sont souvent des informations importantes qu'on aimerait protéger d'une lecture ou d'une écriture non autorisée. C'est pour cette raison que le problème principal dans le contrôle d'accès est le problème de la protection. Il s'agit dans ce problème d'être capable de dire, pour un droit donné, si les commandes du système d'exploitation

permettent d'accéder, à partir d'un état donné, à un état dit non sûr pour le droit en question. Dans une classe particulière de systèmes, ce problème peut être vu comme un problème de planification où l'état du monde est un état de protection, où les opérateurs du planificateur sont des commandes, où l'état but est un état de protection non sûr, et où le plan solution est une séquence de commandes.

Plusieurs travaux de recherche ont été réalisés pour modéliser les systèmes d'exploitation et pour donner des résultats concernant la décidabilité du problème de la protection. Parmi ces travaux ils existent différentes approches pour la modélisation des systèmes d'exploitations, notamment celles basées sur les matrices d'accès comme le modèle de Lampson [17], le modèle de Harrison, Ruzzo et Ullman [14], le modèle de la matrice d'accès typée de Sandhu [20], le modèle de Bell et LaPadula [5], le modèle de Biba [6] et celles basées sur la réécriture de graphe comme le modèle take-grant [16], le modèle grammatical [23]. Le modèle introduit par Harrison, Ruzzo et Ullman est un modèle général pour lequel le problème de la protection est indécidable. Quand aux modèles take-grant et au modèle grammatical, ils constituent une version simplifiée des systèmes d'exploitation. Ils sont donc moins généraux et moins expressifs que le modèle de Harrison, Ruzzo et Ullman. Les modèles de Bell et LaPadula et le modèle de Biba prennent respectivement en considération les propriétés de la confidentialité et de l'intégrité mais ils ne donnent aucun résultat concernant le problème de la protection. Aucun des modèles existants ne prend en considération les notions déontiques comme l'obligation, la recommandation. Ils ne considèrent pas non plus la propriété de la disponibilité.

Notre travail s'inscrit dans le cadre de la sécurité informatique, et plus précisément dans le domaine du contrôle d'accès. Il s'appuie sur le modèle introduit par Harrison, Ruzzo et Ullman. Nous proposons dans ce rapport une extension de ce modèle qui nous permettra de prendre en considération le concept de la disponibilité. Pour cela nous avons introduit la notion de systèmes de protection temporisée qui est indispensable pour gérer le temps et la dépendance temporelle entre événements. Dans le cadre général des systèmes de protection temporisée, nous n'avons pas pu donner de résultats concernant le problème de la protection. C'est pourquoi nous avons considéré une restriction de ces systèmes : les systèmes de protection temporisée spécifiques, pour lesquels nous avons réussi à établir des résultats mathématiques concernant la décidabilité de trois problèmes différents que nous avons appelés problèmes de la protection temporisée.

Ce rapport est composé de quatre chapitres. Dans le premier chapitre, nous citons les différentes approches qui existent pour modéliser la protection. Ensuite,

dans le deuxième chapitre, nous décrivons le modèle de protection introduit par Harrison, Ruzzo et Ullman ainsi que les résultats de décidabilité du problème de la protection. Le troisième chapitre est consacré aux systèmes de protection temporisée, aux trois problèmes de la disponibilité et aux trois problèmes de la protection temporisée. Le quatrième et dernier chapitre concerne les systèmes de protection temporisée spécifiques.

Chapitre 1

Approches existantes pour modéliser la protection

1.1 Introduction

Un système informatique est constitué d'un ensemble de sujets comme les êtres humains et d'un ensemble d'objets comme les fichiers. Le but d'un mécanisme de contrôle d'accès est de gérer tous les accès des sujets aux objets ainsi que la modification de l'état de protection. A travers le mécanisme de contrôle d'accès, le système informatique vérifie l'identité de chaque sujet pour en déduire les droits qu'il a sur chaque objet.
Dans ce chapitre nous allons introduire les concepts de bases et quelques unes des différentes approches existantes [7], pour modéliser la protection dans les systèmes informatiques :

- le modèle "take-grant" ;
- le modèle grammatical ;
- le modèle de Bell et LaPadula ;
- le modèle de Biba.

Ce qui distingue les modèles de Bell et LaPadula et de Biba des autres, c'est leur capacité à traiter les concepts de confidentialité et d'intégrité.
Il existe aussi un modèle important, celui d'Harrison, Ruzzo et Ullman (HRU), qui est plus général que le modèle grammatical et pour lequel nous avons préféré consacrer le chapitre suivant, car notre travail s'appuie sur ce modèle.

Nous allons concidérer ci-dessous, les deux familles principales d'approches du controle d'accès : les approches basées sur la réécriture de graphe [16, 10] et les

approches basées sur la matrice des droits d'accès [5, 6].

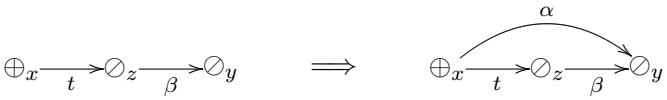
1.2 Approches basées sur la réécriture de graphe

1.2.1 le modèle "take-grant"

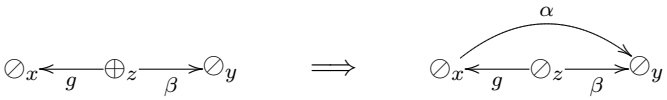
Le modèle de protection "take-grant", introduit par Jones, Lipton et Snyder [16], représente un état de protection sous la forme d'un graphe orienté et étiqueté. Les sommets de ce graphe sont des sujets (représentés par $\oplus$) ou des objets (représentés par $\otimes$). Les sommets qui peuvent être des sujets ou des objets sont représentés par ($\oslash$). Les arcs sont étiquetés par des ensembles de droits. Les droits sont des éléments de l'ensemble $\{t, g, r, w\}$, où t est un droit permettant de prendre d'autres droits, g est un droit permettant d'accorder d'autres droits, r est le droit de lecture et w est le droit d'écriture. Par exemple, si l'étiquette de l'arc allant du sujet x à l'objet y contient le droit r, cela signifie que le sujet x a le droit de lire l'objet y.

Le système de protection est défini par quatre règles de réécriture du graphe, qui permettent de le modifier et par conséquent de modifier l'état de protection du système.

1. La règle "**take**" : Soit x, y et z trois sommets distincts dans un graphe de protection G_0, où x est un sujet. S'il existe un arc de x à z étiqueté par γ avec $t \in \gamma$ et s'il existe un arc de z à y étiqueté par β alors un nouveau graphe G_1 est obtenu de G_0 en ajoutant un arc de x à y étiqueté par α, $\alpha \subseteq \beta$.
   Graphiquement :

   On dit que x prend de z les droits α sur y.
2. La règle "**grant**" : Soit x, y et z trois sommets distincts dans un graphe de protection G_0, où z est un sujet. S'il existe un arc de z à x étiqueté par γ avec $g \in \gamma$ et s'il existe un arc de z à y étiqueté par β alors un nouveau graphe G_1 est obtenu de G_0 en ajoutant un arc de x à y étiqueté par α, $\alpha \subseteq \beta$.
   Graphiquement :

   On dit que z accorde à x les droits α sur y

3. La règle "**créer**" : Soit x un sommet de type sujet dans un graphe de protection G_0 et soit β un ensemble de droits. Cette règle permet de définir un nouveau graphe G_1 en ajoutant au graphe G_0 un nouveau sommet y et un arc de x à y étiqueté par β.
   Graphiquement :
   $$\oplus_x \quad \Longrightarrow \quad \oplus_x \xrightarrow{\alpha} \oslash_y$$
   On dit que x a créé un nouveau sujet (ou un nouvel objet) sur lequel il a les droits β.
4. La règle "**supprimer**" : Soit x et y deux sommets distincts dans un graphe de protection G_0 tels que x est un sujet possédant les droits β sur y. Si α est un ensemble de droits alors cette règle permet de définir un nouveau graphe G_1 en supprimant de β les droits de α.
   Graphiquement :
   $$\oplus_x \xrightarrow[\beta]{} \oslash_y \quad \Longrightarrow \quad \oplus_x \xrightarrow{\beta-\alpha} \oslash_y$$
   On dit que x se prive des droits α sur y.

Sécurité pour le modèle "take-grant"

Pour le modèle "take-grant", le problème de la protection est le problème de décision suivant :

Entrée : un graphe orienté et étiqueté G, un ensemble de droits α et deux sommets x et y.

Sortie : déterminer s'il existe, une séquence de graphe $G_1, G_2, \ldots, G_n$ telle que $G = G_1 \rightarrow G_2 \rightarrow, \ldots, \rightarrow G_n$ et s'il existe dans G_n un arc de x à y étiqueté par α.

Ce problème de décision, ainsi que le démontre Jones, Lipton et Snyder [16], est traitable en temps linéaire. Cette faible complexité, s'explique par le fait que les règles de réécriture, qui permettent de modifier l'état de protection, sont fixes et ne font pas partie des entrées du problème.

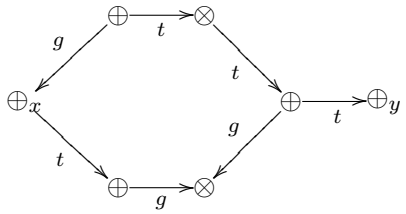
1.2.2 Le modèle grammatical

Comme pour le modèle de protection "take-grant", le modèle grammatical de protection, introduit par Budd et Lipton [10], représente un état de protection sous la forme d'un graphe orienté et étiqueté. A ce graphe est associé un ensemble fini

C de mots défini à travers une alphabet. Chaque mot de l'ensemble C représente un chemin dans ce graphe. Les lettres à partir desquelles les mots sont formés appartiennent à l'ensemble $T \times V \times T$, où T est l'ensemble des types de sommets et V l'ensemble de droits avec l'orientation de l'arc.

Exemple 1

Dans le modèle "take-grant" :
- les deux types de sommets sont les types sujet et objet ; ils sont représentés par $T = \{S, O\}$;
- les étiquettes des arcs avec leur orientation $V = \{t^{\rightarrow}, g^{\rightarrow}, r^{\rightarrow}, w^{\rightarrow}, t^{\leftarrow}, g^{\leftarrow}, r^{\leftarrow}, w^{\leftarrow}\}$;
- $[s, t^{\rightarrow}, s]$ est la lettre qui code $\oplus \xrightarrow{t} \oplus$;
- $[s, g^{\leftarrow}, o]$ code $\oplus \xleftarrow{g} \otimes$;
- le graphe G_1 :



définit les deux mots suivants :

$$[s, g^{\leftarrow}, s][s, t^{\rightarrow}, o][o, t^{\rightarrow}, s][s, t^{\rightarrow}, s] \quad (1)$$

$$[s, t^{\rightarrow}, s][s, g^{\rightarrow}, o][o, g^{\leftarrow}, s][s, t^{\leftarrow}, s].$$

Le système de protection est défini par une grammaire indépendante du contexte où les règles de production sont de la forme :

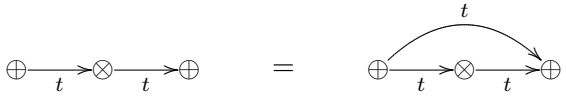
$$S_1 \rightarrow S'_1 \dots S'_n$$

où S_1, S'_i, $i \in \{1, \dots, n\}$ sont des éléments de $T \times V \times T$.

Exemple 2

Considérons encore le modèle "take-grant"

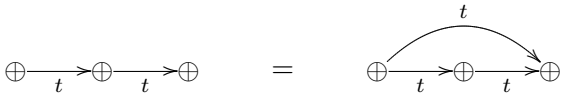
– La règle “**take**” :



est exprimée par la production grammaticale :

$[s, t^{\rightarrow}, s] \longrightarrow [s, t^{\rightarrow}, o][o, t^{\rightarrow}, s]$ (2)

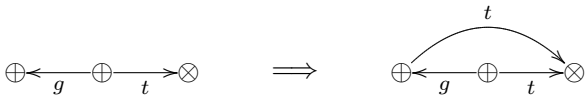
– Une autre version de la règle “**take**” :



est exprimée par la production grammaticale :

$[s, t^{\rightarrow}, s] \longrightarrow [s, t^{\rightarrow}, s][s, t^{\rightarrow}, s]$ (3)

– Même chose pour la règle “**grant**” :



qui est exprimée par la production grammaticale :

$[s, t^{\rightarrow}, o] \longrightarrow [s, g^{\leftarrow}, s][s, t^{\rightarrow}, o]$ (4)

Le mot (1) peut être produit grâce aux règles (2), (3) et (4) comme suit :

$$[s, t^{\rightarrow}, s] \longrightarrow_{(3)} [s, t^{\rightarrow}, s][s, t^{\rightarrow}, s] \longrightarrow_{(2)} [s, t^{\rightarrow}, o][o, t^{\rightarrow}, s][s, t^{\rightarrow}, s] \longrightarrow_{(4)} [s, g^{\leftarrow}, s][s, t^{\rightarrow}, o][o, t^{\rightarrow}, s][s, t^{\rightarrow}, s]$$

L’ensemble complet de règles est donné par [9].

Sécurité pour le modèle grammatical

Pour le modèle grammatical, le problème de la protection est le problème de décision suivant :

Entrée : un graphe orienté et étiqueté G, une grammaire L, un élément $v \in V$ et deux sommets x et y.

Sortie : déterminer si $C_{xy} \cap L([f(x), v, f(y)]) \neq \emptyset$.

où :
- C_{xy} est l'ensemble des mots associé à l'ensemble de tous les chemins entre le sommet x et le sommet y dans le graphe G ;
- Z : ensemble de sommets de G ;
- $f : Z \rightarrow T$, une fonction qui appliquée à un sommet de G donne son type ;
- $L(u), u \in T \times V \times T$, est l'ensemble de mots produits à partir de la lettre u, dans la grammaire L.

Exemple 3

Si on considère, l'instance du problème de la protection suivante : le graphe G_1 de l'exemple 1, une grammaire dans laquelle les règles de production sont les règles de réécritures du modèle "take-grant", le droit avec l'orientation de l'arc $t^{\rightarrow}$ et les sommets x et y du graphe G_1, alors $C_{xy} \cap L([s, t^{\rightarrow}, s]) \neq \emptyset$.

Preuve: Dans l'exemple 1 le mot (1) est associé à un chemin entre x et y dans G_1. Donc il appartient bien à l'ensemble C_{xy}. Nous avons montré dans l'exemple 2 que le mot (1) peut être produit à partir de la lettre $[s, t^{\rightarrow}, s]$ en appliquant les règles de réécriture (2), (3) et (4). Donc il appartient aussi à $L([s, t^{\rightarrow}, s])$. D'où $C_{xy} \cap L([s, t^{\rightarrow}, s]) \neq \emptyset$. $\dashv$

Budd et Lipton montre que ce problème de décision est traitable en temps polynomial $O(n^3)$.

1.3 Approches basées sur la matrice des droits d'accès

Les approches considérées dans cette section, sont basées sur la notion de matrice d'accès, introduite par Lampson [17]. Elles se basent sur l'idée d'associer à chaque sujet s et chaque objet o d'un système de protection un ensemble $M[s, o]$ de droits. La relation $r \in M[s, o]$ s'interprète : "Le sujet s à le droit r sur l'objet o". M est appelée matrice d'accès. Cette approche est utilisée dans le modèle de Bell et LaPadula [5] ainsi que dans le modèle de Biba [6].

1.3.1 Le modèle de Bell et LaPadula

La confidentialité est une des propriétés les plus importantes de la sécurité informatique. Une information est dite confidentielle pour un ensemble d'utilisateurs lorsqu'ils ne peuvent rien connaître de cette information. Le modèle de Bell et LaPadula (BLP) est un modèle qui s'intéresse principalement à la confidentialité. Il a

Niveaux de sécurité	Utilisateurs	Fichiers
Top secret	Maurice, Louise	Fichiers personnels
Secret	Thomas	Fichiers du mél
Confidentiel	Léonie	Fichiers des comptes-rendus d'activité
Non classé	François	Fichiers des appels téléphoniques

TAB. 1.1 – Fonction F

M	Fichiers personnels	Fichiers du mél	Fichiers des comptes-rendus d'activité	Fichiers des appels téléphoniques
François	$read$	$write$		$read$
Léonie		$write$	$read$	$read$
Louise	$read$	$write$	$read$	$write$
Maurice	$read, write$	$read$	$read$	$read$
Thomas		$read$	$write$	

TAB. 1.2 – Matrice d'accès M

été élaboré par Bell est LaPadula en 1975 [5] pour le département américain de la défense. Dans le modèle BLP les ensembles S et O ne sont pas modifiés et l'ensemble des droits contient les droits $read$ et $write$. Dans ce modèle on se donne un ensemble NS de niveaux de sécurité et une fonction $F : S \cup O \rightarrow NS$ qui, appliquée à un sujet ou à un objet dans un état de protection, donne le niveau de sécurité de son argument dans ce même état. L'ensemble NS est muni d'un ordre total "$<$". Si $l, l' \in NS$ sont tels que $l < l'$, cela signifie que le niveau de sécurité l est moins sensible que le niveau l'. Les niveaux de sécurité généralement considérés sont les éléments de l'ensemble NS={top secret, secret, confidentiel, non classé}. Dans ce modèle un état est un couple de la forme (F, M), où M est une matrice d'accès et où F est une fonction de la forme précédente. Exemple, l'état défini par la fonction F de la table 1.1 et la matrice M de la table 1.2.

Dans un état de protection (F, M), le sujet s peut lire l'objet o ssi $read \in M[s, o]$ et $F(s) \geq F(o)$. Dans (F, M), s peut écrire dans o ssi $write \in M[s, o]$ et $F(o) \geq F(s)$.

Dans l'exemple précédent, l'utilisateur "François" ne peut pas lire le fichier "Fichiers personnels" car le niveau de sécurité de "Fichiers personnels" et supérieur à celui de "François". L'utilisateur "Louis" ne peut pas écrire dans le fichier "Fichiers du mél" car le niveau de sécurité de "Louis" et supérieur à celui de "Fichiers du mél". Si on laissait "Louis" écrire dans "Fichiers du mél", il risquerait d'écrire des informations top secrètes que l'utilisateur "Thomas" pourra lire.

Aujourd'hui encore, le modèle BLP reste un modèle de sécurité de référence. Lorsqu'un nouveau modèle de confidentialité est proposé, on le compare très souvent à BLP.

1.3.2 Le modèle de Biba

Comme la confidentialité, l'intégrité est une des propriétés les plus importantes de la sécurité informatique. Une information est dite intègre pour un ensemble d'utilisateurs, lorsqu'ils peuvent se fier à elle. Biba en 1977 [6] a proposé un modèle destiné au traitement de l'intégrité des objets, dans le domaine de la finance. Ce modèle est le dual mathématique du modèle de Bell et LaPadula.

Comme pour le modèle de Bell et LaPadula, dans le modèle de Biba, les ensembles S et O ne sont pas modifiés et l'ensemble des droits contient les droits $read$ et $write$. Dans ce modèle on se donne un ensemble NF de niveaux de fiabilité et une fonction $H : S \cup O \rightarrow NF$ qui, appliquée à un sujet ou à un objet dans un état de protection, donne le niveau de fiabilité de son argument dans ce même état. L'ensemble NF est muni d'un ordre total "$<$". Si $n, n' \in NF$ sont tels que $n < n'$, cela signifie que le niveau n est moins fiable que le niveau n'. Dans ce modèle un état est un couple de la forme (H, M), où M est une matrice d'accès et où H est une fonction de la forme précédente. Dans un état de protection (H, M), le sujet s peut lire l'objet o ssi $read \in M[s, o]$ et $H(s) \leq H(o)$. Dans (H, M), s peut écrire dans o ssi $write \in M[s, o]$ et $H(o) \leq H(s)$.

Dans le modèle de Biba, un utilisateur ne peut pas écrire dans un fichier de plus grand niveau de confiance, cela pour éviter que l'utilisateur y introduise des données fausses et incorrectes. De même, un utilisateur ne peut pas lire dans un fichier de niveau de fiabilité inférieur, cela pour éviter que l'utilisateur reçoive des informations fausses et incorrectes.

Chapitre 2

Une approche uniforme pour modéliser la protection : le modèle HRU

2.1 Introduction

Les systèmes informatiques modernes contiennent des informations importantes. Un accès non autorisé aux données en lecture ou en écriture peut créer des problèmes significatifs. Harrison, Ruzzo et Ullman se sont intéressés en 1976 à la protection et à la sécurité des systèmes d'exploitation, en proposant une approche uniforme pour modéliser la protection de sorte que le modèle soit général et qu'il capture l'essence de la sécurité dans le contrôle d'accès.

2.2 Le modèle HRU

Dans le model HRU, Harrison, Ruzzo et Ullman considèrent qu'un système informatique contient une collection d'objets abstraits dont la sécurité est importante. Dans la pratique ces objets représentent des fichiers contenant des données importantes. Ils considèrent aussi que dans ce système il existe plusieurs utilisateurs ou processus qui peuvent avoir des droits d'accès sur ces fichiers.

P	s_1	s_2	o_1
s_1	r_1, r_2	r_0, r_1, r_2	r_1
s_2	r_0, r_1, r_2	r_1, r_2	r_2

TAB. 2.1 – Etat de protection Δ.

2.2.1 Etats de protection

Soit SC un ensemble dénombrable de constantes de type sujet où chaque élément est noté s, s' etc, avec éventuellement des indices, OC un ensemble dénombrable de constantes de type objet où chaque élément est noté o, o' etc, avec éventuellement des indices et R un ensemble fini de droits de cardinalité $|R|$ où chaque élément est noté r, r' etc, avec éventuellement des indices.
Partant du principe que les processus peuvent à la fois être considérés comme des entités actives et des entités passives du système, Harrison, Ruzzo et Ullman considèrent toujours que tous les sujets sont aussi des objets : $SC \subseteq OC$. Un état de protection est un triplet (S, O, P) où $S \subseteq SC$ est un ensemble fini de sujets de cardinalité $\mid S \mid$, $O \subseteq OC$ est un ensemble fini d'objets de cardinalité $\mid O \mid$, $S \subseteq O$, et P une matrice d'accès qui, pour chaque sujet $s \in S$ et pour chaque objet $o \in O$, définit l'ensemble $P[s, o] \subseteq R$ des droits que le sujet s a sur l'objet o. Les états de protection seront notés par les lettres Δ, Δ', etc, avec éventuellement des indices. la table 2.1 illustre un état de protection Δ présenté sous la forme d'une matrice. Les éléments de la matrice spécifient les droits que chaque sujet a sur chaque objet.

2.2.2 Opérations primitives

Soit SV un ensemble dénombrable de variables de type sujet où chaque élément est noté x, x' etc, avec éventuellement des indices et OV un ensemble dénombrable de variables de type objet où chaque élément est noté y, y' etc, avec éventuellement des indices. Nous supposons que $SV \subseteq OV$. Il existe six opérations primitives permettant de modifier l'état de protection :

- "créer le sujet x" ;
- "détruire le sujet x ;
- "créer l'objet y" ;

– “détruire l’objet y” ;

– “introduire r dans $P(x, y)$” ;

– “supprimer r de $P(x, y)$”.

Les opérations primitives seront notées par les lettres π, π', etc, avec éventuellement des indices.

2.2.3 Transition entre états

Les assignations remplacent les variables par des constantes. Ce sont donc des ensembles finis de la forme : $\{x_1/s_1, \ldots, x_k/s_k, y_1/o_1, \ldots, y_l/o_l\}$ où pour tout entier $i \in \{1, \ldots, k\}$ et pour tout entier $j \in \{1, \ldots, l\}$, x_i est une variable de type sujet, s_i une constante de type sujet, y_j une variable de type objet, et o_j une constante de type objet. Bien-sûr nous ne considérons que des assignations dans lesquelles les variables $x_1, \ldots, x_k$ (respectivement $y_1, \ldots, y_l$) sont deux à deux distinctes. Les assignations seront notées par les lettres θ, θ', etc, avec éventuellement des indices.
Un état de protection Δ peut être modifié grâce aux opérations primitives relativement à une assignation. Soit θ une assignation et π une opération primitive. Supposons qu’il n’existe aucune variable dans $\theta(\pi)$, c’est-à-dire que chaque variable dans π est remplacée par une constante en utilisant θ. Si $\Delta = (S, O, P)$ et $\Delta' = (S', O', P')$ sont des états de protection, Δ' est dit dérivable de Δ en utilisant θ et π, représenté par $\Delta \longrightarrow_{\pi}^{\theta} \Delta'$, ssi une des conditions suivantes est satisfaite :

– π est “créer le sujet x”, $\theta(x)$ n’appartient pas à O, $S' = S \cup \{\theta(x)\}$, $O' = O \cup \{\theta(x)\}$, $P'[s, o] = P[s, o]$ pour tout $(s, o) \in S \times O$, $P'[\theta(x), o] = \emptyset$ pour tout $o \in O'$ et $P'[s, \theta(x)] = \emptyset$ pour tout $s \in S'$.

– π est “détruire le sujet x”, $\theta(x)$ appartient à S, $S' = S \setminus \{\theta(x)\}$, $O' = O \setminus \{\theta(x)\}$ et $P'[s, o] = P[s, o]$ pour tout $(s, o) \in S' \times O'$.

– π est “créer l’objet y”, $\theta(y)$ n’appartient pas à O, $S' = S$, $O' = O \cup \{\theta(y)\}$, $P'[s, o] = P[s, o]$ pour tout $(s, o) \in S \times O$ et $P'[s, \theta(y)] = \emptyset$ pour tout $s \in S'$.

– π est “détruire l’objet y”, $\theta(y)$ appartient à $O \setminus S$, $S' = S$, $O' = O \setminus \{\theta(y)\}$ et $P'[s, o] = P[s, o]$ pour tout $(s, o) \in S' \times O'$.

– π est "introduire r dans $P(x, y)$", $\theta(x)$ appartient à S, $\theta(y)$ appartient à O, $S' = S$, $O' = O$ et pour tout $(s, o) \in S \times O$:

$$P'[s, o] = \begin{cases} P[s, o] \cup \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ P[s, o] & \text{sinon} \end{cases}$$

– π est "supprimer r de $p(x, y)$", $\theta(x)$ appartient à S, $\theta(y)$ appartient à O, $S' = S$, $O' = O$ et pour tout $(s, o) \in S \times O$:

$$P'[s, o] = \begin{cases} P[s, o] \setminus \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ P[s, o] & \text{sinon} \end{cases}$$

2.2.4 Système de protection

Dans le modèle HRU les opérations primitives sont appliquées indirectement par l'intermédiaire de commandes de la forme :

– "si C_1 et ... et C_i alors début π_1 ; ... ; π_j fin",

où $C_1, \ldots, C_i$ sont des conditions élémentaires de la forme $r \in P[x, y]$ et π_1, ..., π_j sont des opérations primitives. Le nombre de ces conditions est l'entier non négatif i, et le nombre de ces opérations primitives est l'entier positif j. Les commandes sont appliquées en remplaçant toutes les variables de la commande par des constantes. Ensuite, si les conditions initiales $C_1, \ldots, C_i$ sont évaluées à vrai dans l'état de protection courant alors les opérations primitives $\pi_1, \ldots, \pi_j$ sont séquentiellement éxécutées. Les commandes seront notées par les lettres α, α', etc, avec éventuellement des indices.

Un système de protection est un ensemble fini $\{\alpha_1, \ldots, \alpha_k\}$ de commandes et un ensemble fini R de droits.

Soit θ une assignation et α une commande de la forme :"si C_1 et ... et C_i alors début π_1 ; ... ; π_j fin". Supposons qu'il n'existe aucune variable dans $\theta(\alpha)$, c'est-à-dire que chaque variable dans α est remplacée par une constante en utilisant θ. Si $\Delta = (S, O, P)$ et $\Delta' = (S', O', P')$ sont des états de protection alors Δ' est dit dérivable de Δ en utilisant θ et α, représenté par $\Delta \longrightarrow_{\alpha}^{\theta} \Delta'$, ssi une des deux conditions suivantes est satisfaite :

1. Si les conditions de α ne sont pas satisfaites alors $\Delta = \Delta'$.
2. Sinon il existent des états de protections $\Delta_0, \Delta_1, \ldots, \Delta_j$ tels que

$$\Delta = \Delta_0 \longrightarrow_{\pi_1}^{\theta} \Delta_1 \longrightarrow_{\pi_2}^{\theta} \ldots, \longrightarrow_{\pi_j}^{\theta} \Delta_j = \Delta'$$

Nous dirons que $\Delta \longrightarrow_\alpha \Delta'$ lorsqu'il existe une assignation θ telle que $\Delta \longrightarrow_\alpha^\theta \Delta'$. Nous dirons que $\Delta \longrightarrow_\Pi \Delta'$ lorsqu'il existe une commande $\alpha \in \Pi$ telle que $\Delta \longrightarrow_\alpha \Delta'$. Il conviendra de noter $\Delta \longrightarrow_\Pi^* \Delta'$ lorsqu'il existe un entier non négatif n et des états de protection $\Delta_0, \Delta_1, \ldots, \Delta_n$ tels que

$$\Delta = \Delta_0 \longrightarrow_\Pi \Delta_1 \longrightarrow_\Pi \ldots \longrightarrow_\Pi \Delta_n = \Delta'.$$

Un système de protection est dit monotone ssi aucune de ses commandes ne contient une opération primitive de la forme "détruire" ou "supprimer". Il est monoconditionnel ssi aucune de ses commandes ne contient plus d'une condition tandis qu'il est mono-operationnel ssi aucune de ses commandes ne contient plus d'une opération primitive. Les systèmes de protection seront notés par les lettres Π, Π', etc, avec éventuellement des indices.

2.3 Sécurité

Il est important d'être capable de dire avec précision si un système de protection est sûr ou pas. Pour dire qu'un système n'est pas sûr il faut d'abord définir ce qu'est l'apparition d'un droit.

Soit Π un système de protection, On dit que la commande $\alpha \in \Pi$ de la forme :"si C_1 et $\ldots$ et C_i alors début π_1 ; $\ldots$; π_j fin", fait apparaître le droit r dans l'état de protection $\Delta = (S, O, P)$, si α permet d'introduire r dans une cellule de la matrice P qui ne le contenait pas. Formellement, α permet d'introduire r dans P lorsqu'il existe une assignation θ, telle que $\theta(\alpha)$ ne contient aucune variable et que :

1. θ évalue à vrai toutes les conditions de α.
2. Il existe $1 \leq k \leq j$ et des états de protections $\Delta_0, \Delta_1, \ldots, \Delta_j = (S', O', P')$ avec $\Delta_{k-1} = (S_{k-1}, O_{k-1}, P_{k-1})$ et $\Delta_k = (S_k, O_k, P_k)$ tels que :
   $$\Delta = \Delta_0 \longrightarrow_{\pi_1}^\theta \Delta_1 \longrightarrow_{\pi_2}^\theta \ldots, \longrightarrow_{\pi_j}^\theta \Delta_j = \Delta'$$
   et il existe un sujet s et un objet o tels que $s \in S_k$, $o \in O_k$ et $r \in P_k[s, o]$, mais $s \in S_{k-1}$, $o \in O_{k-1}$ et $r \notin P_{k-1}[s, o]$.

Soit Π un système de protection, nous dirons qu'il est non sûr pour un droit r relativement à un état de protection initial Δ_0, lorsqu'il existe un état de protection Δ et une commande α tels que :

1. $\Delta_0 \longrightarrow_\Pi^* \Delta$.

2. α fait apparaître r dans Δ.

Pour le modèle HRU, le problème de la protection est le problème de décision suivant :

Entrée : un système de protection Π, un état de protection Δ et un droit r.

Sortie : déterminer si Π est non sûr pour le droit r relativement à Δ.

Harrison, Ruzzo et Ullman montrent [14] que le problème de l'arrêt des machines de turing est réductible au problème de la protection d'où :

Théorème 1 *Il est indécidable de savoir si un système de protection est non sûr, pour un droit r relativement à un état de protection Δ.*

Dans le cas général le problème de la protection est indécidable ainsi qu'on vient de le voir. Mais si on se restreint à une classe particulière de systèmes de protection le problème devient décidable. C'est ce qu'ont montré Lipton et Snyder pour la classe des systèmes de protection sans commandes de la forme "créer le sujet x", Harrison Ruzzo et Ullman pour la classe des systèmes de protection mono-opérationnels et Harrison et Ruzzo pour la classe des systèmes de protection monotones monoconditionnels.

Théorème 2 *Il existe un algorithme qui décide si un système de protection sans commandes de la forme ''créer le sujet x", est non sûr pour un droit r relativement à un état de protection Δ.*

Pour prouver ce théorème Lipton et Snyder [18] montrent que le problème de la protection pour la classe des systèmes de protection sans commandes de la forme ''créer le sujet x", est équivalent à un problème de recouvrement dans les systèmes d'addition de vecteurs.

Théorème 3 *Il existe un algorithme qui décide si un système de protection mono-opérationnel est non sûr pour un droit r relativement à un état de protection Δ.*

Harrison, Ruzzo et Ullman proposent, une démonstration de ce théorème. Cette démonstration est incorrecte. Nous allons voir pourquoi.

Soit Π un système de protection mono-opérationnel. On veut savoir s'il est non sûr pour le droit r relativement à un état de protection $\Delta = (S, O, P)$ donné. Dans la démonstration proposée par Harrison, Ruzzo et Ullman [14], les auteurs montrent qu'une séquence $\Delta = \Delta_0 \longrightarrow_\Pi \Delta_1 \longrightarrow_\Pi \ldots \longrightarrow_\Pi \Delta_n = \Delta'$ de longueur minimale et telle qu'une commande de Π fait apparaître r dans Δ', est forcément telle que :

– aucune commande de la forme "supprimer" n'est utilisée par cette séquence ;
– si $|S| > 0$ alors aucune commande de la forme "créer" n'est utilisée par cette séquence.

Remarque

Le système de protection étant mono-opérationnel la forme d'une commande est égale à la forme de l'unique opération primitive qu'elle contient.

Considérons le cas où $\Delta = (S, O, P)$ est tel que $|S| > 0$ et où pour tout sujet $s \in S$ et pour tout objet $o \in O$ on a $P[s, o] = \{r\}$. Soit (s', o') où $s' \in S_n$ et $o' \in O_n$, la cellule dans laquelle le droit r apparaît. Pour Harrison, Ruzzo et Ullman, il n'existe dans la séquence aucune commande de la forme "supprimer" ou "créer". Donc $s' \in S_0$, $o' \in O_0$, $r \in P_0[s', o']$ et $r \in P_n[s', o']$. Le droit r ne peut pas donc apparaître dans la cellule (s', o') ce qui est absurde. Par conséquent la séquence minimale doit contenir soit une commande de la forme "créer" pour faire apparaître r dans une nouvelle cellule ou bien une commande de la forme "supprimer r" pour faire apparaître r dans la cellule où il a été supprimé.

Nous avons ainsi trouvé un contre-exemple à la démonstration de la décidabilité du problème de la protection dans le cas des systèmes de protection mono-opérationnels, C'est pourquoi nous proposons une preuve du théorème 3 dans l'annexe A.

Harrison et Ruzzo montrent [12] que le problème de la correspondance de Post est réductible au problème de la protection dans la classe des systèmes de protection monotones d'où :

Théorème 4 *Il est indécidable de savoir si un système de protection monotone est non sûr pour un droit r relativement à un état de protection Δ.*

Ils ont aussi proposé une démonstration, dans laquelle ils montrent qu'en restreignant encore plus la classe des systèmes de protection et en considérant les systèmes non seulement monotones mais aussi monoconditionnels, le problème de la protection devient alors décidable.

Théorème 5 *Il existe un algorithme qui décide si un système de protection monotone monoconditionnel est non sûr pour un droit r relativement à un état de protection Δ.*

P_0	s_1	s_2
s_1	r, r_1, r_2	r, r_1, r_2
s_2	r, r_1, r_2	r, r_1, r_2

TAB. 2.2 – Etat de protection Δ.

P_0	s_1	s_2	o_1
s_1	r, r_1, r_2	r, r_1, r_2	
s_2	r, r_1, r_2	r, r_1, r_2	

TAB. 2.3 – Etat de protection Δ_1.

La démonstration que proposent Harrison et Ruzzo de ce théorème est incorrecte. Nous allons voir pourquoi.

Soit Π un système de protection monotone monoconditionnel. On veut savoir s'il est non sûr pour le droit r relativement à un état de protection $\Delta = (S, O, P)$ donné. Dans la démonstration proposée par Harrison et Ruzzo [12], les auteurs montrent qu'une séquence de longueur minimale

$$\Delta = \Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_n}_{\alpha_n} \Delta_n \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} \Delta_{n+1}$$

où $\theta_1, \ldots \theta_{n+1}$ sont des assignations, $\alpha_1, \ldots, \alpha_{n+1} \in \Pi$ et telle que α_{n+1} fait apparaître r dans Δ_n, est forcément telle que pour tout entier $l \in \{1 \ldots n+1\}$:

– si "$r_l \in P[x, y]$" est la condition de la commande α_l alors on a $r_l \in P_{l-1}[\theta_l(x), \theta_l(y)]$ et ($r_l \notin P_{l-2}[\theta_l(x), \theta_l(y)]$ ou $\theta_l(x) \notin S_{l-2}$ ou $\theta_l(y) \notin O_{l-2}$) avec $\theta_l(x) = s_l$ et $\theta_l(y) = o_l$.

Considérons le cas où Π est le système de protection monotone monoconditionnel défini par l'ensemble de droits $\{r, r_1, r_2\}$ et l'ensemble de commandes :

– α : "début introduire r_1 dans $P[x, y]$; introduire r_2 dans $P[x', y']$ fin" ;
– α' : "si $r_1 \in P[x, y]$ alors créer l'objet y'" ;
– α'' : "si $r_2 \in P[x, y]$ alors introduire r dans $P[x, y']$".

et où $\Delta = (S, O, P)$ est l'état de protection représenté par la table 2.2. Dans l'état de protection Δ le droit r appartient à toute les cellules et le système de protection Π ne contient aucune commande de la forme "supprimer". Donc pour faire apparaître r dans Δ_n il faut créer au moins une nouvelle cellule pour y introduire le droit r. La séquence minimale dans ce cas doit contenir au moins les commandes α' et α''. De ce fait elle ne peut être que la séquence

$$\Delta = \Delta_0 \longrightarrow^{\theta}_{\alpha'} \Delta_1 \longrightarrow^{\theta}_{\alpha''} \Delta_2$$

P_0	s_1	s_2	o_1
s_1	r, r_1, r_2	r, r_1, r_2	r
s_2	r, r_1, r_2	r, r_1, r_2	

TAB. 2.4 – Etat de protection Δ_2.

où l'assignation $\theta = \{x/s_1, y/s_1, y'/o_1\}$ et où Δ_1 et Δ_2 sont respectivement les états de protection représentés par la table 2.3 et la table 2.4. Cette séquence ne satisfait pas les conditions de Harrison et Ruzzo car pour $i = 2$, $r_2 \in P_{i-1}[s_1, s_1]$.

Nous avons ainsi trouvé un contre-exemple à la démonstration de la décidabilité du problème de la protection dans le cas des systèmes de protection monotones monoconditionnels. Nous n'avons pas pu donner de preuve pour le théorème 4 dans le cas où l'ensemble des sujets est inclus dans celui des objets, car nous n'avons pas eu le temps de considérer tous les cas qui doivent être traités. C'est pour cette raison que nous proposons une preuve dans l'annexe B en considérant que l'ensemble des sujets est disjoint de celui des objets. D'ailleur cette hypothèse est souvent considérée par Sandhu [21].

Aprés ce tour d'horizon sur le modèle HRU et sur la décidabilité du problème de la protecion, nous citons ci-dessous quelques variantes de ce modèle.

Le modèle HRU avec des conditions négatives [22, 4] est utilisé dans le cas par exemple, où deux droits spécifiques r_1 et r_2 ne peuvent pas être attribués à un utilisateur sur le même fichier. Ainsi pour introduire l'un d'eux dans une cellule il faut vérifier que l'autre n'y appartient pas.

Le modèle HRU avec des notions déontiques [4] est utilisé pour obliger, interdire, etc un sujet à faire une certaine action, par exemple obliger un utilisateur à lire le mode d'emploi d'un logiciel.

Le modèle avec rôles [1] quant à lui permet d'attribuer des droits aux sujets selon leur rôle. Par exemple dans un hôpital les médecins et les infirmiers n'ont pas les mêmes droits d'accès aux fichiers des patients. Cette modélisation permet de réduire la taille de la matrice d'accès qui est dans la pratique souvent creuse.

Nous proposons dans le chapitre suivant une autre variante du modèle HRU, qui permettra de prendre en considération la propriété de la disponibilité.

Chapitre 3

Systèmes de protection temporisée

3.1 Introduction

La sécurité informatique repose sur la confidentialité, l'intégrité et la disponibilité. On s'intéresse dans ce rapport à la disponibilité dans les systèmes de protection. Cette dernière se réfère à la possibilité d'utiliser l'information ou la ressource désirée. C'est un aspect important de la qualité de conception du système. Un utilisateur peut bloquer l'accès à un fichier en le rendant non disponible, d'où la notion de sûreté et l'importance de la disponibilité dans les système de protection. Le paramètre temporel est indispensable pour définir la disponibilité. C'est pourquoi nous allons introduire le concept de système de protection temporisée qui constitue une nouvelle approche pour la formalisation de systèmes de protection temps-réel où les nombres réels sont utilisés pour représenter le temps.

3.2 Etats de protection

L'ensemble des sujets est l'ensemble des entités actives, comme les êtres humains. L'ensemble des objets est l'ensemble des éléments passifs comme les fichiers. Les droits de notre modèle abstrait correspondent aux droits du système Unix comme : $read$, $write$, etc. Nous présentons le concept d'état de protection pour décrire le lien entre sujets, droits et objets.
Un état de protection (S, O, A, B, P) a cinq composantes :

- un ensemble fini de sujets S de cardinalité $\mid S \mid$;
- un ensemble fini d'objets O de cardinalité $\mid O \mid$;

- une matrice d'actions A qui a une ligne pour chaque sujet de S et une colonne pour chaque objet de O et où $A[s, o], s \in S, o \in O$, définit les actions du sujet s sur l'objet o ;
- une matrice de buts B qui a une ligne pour chaque sujet de S et une colonne pour chaque objet de O et où $B[s, o], s \in S, o \in O$, définit les buts d'actions du sujet s sur l'objet o ;
- une matrice de permissions P qui a une ligne pour chaque sujet de S et une colonne pour chaque objet de O, et où $P[s, o], s \in S, o \in O$ définit les permissions d'actions du sujet s sur l'objet o ;

avec $A \subseteq B \subseteq P$. Si $r \in A[s, o]$, cela signifie que s effectue, dans l'état courant, l'action caratérisée par le droit r ; si $r \in B[s, o]$, cela signifie que s souhaite effectuer l'action caractérisée par le droit r ; si $r \in P[s, o]$ cela signifie que s à le droit d'effectuer l'action caractérisée par le droit r. Nous supposons dans ce chapitre que l'ensemble des sujets est disjoint de celui des objets. En effet, comme Sandhu [21], nous considérons que les sujets sont des êtres humains et que les objets sont des fichiers.

3.3 Opérations primitives

Il existe 10 opérations primitives permettant de modifier l'état de protection :

- "créer le sujet x" ;
- "détruire le sujet x ;
- "créer l'objet y" ;
- "détruire l'objet y" ;
- "introduire r dans $A(x, y)$" ;
- "supprimer r de $A(x, y)$" ;
- "introduire r dans $B(x, y)$" ;
- "supprimer r de $B(x, y)$" ;
- "introduire r dans $P(x, y)$" ;

– “supprimer r de $P(x, y)$”.

Les opérations primitives seront notées par les lettres π, π', etc, avec éventuellement des indices.

3.4 Transition entre états

Comme dans le chapitre précédent les assignations remplacent les variables par des constantes. Elles seront notées par les lettres θ, θ', etc, avec éventuellement des indices. Soit z une variable de type sujet ou objet, on dit que θ' est équivalente à θ relativement à z (noté $\theta' \approx_z \theta$) ssi $\theta'(z') = \theta(z')$ pour toute variable $z' \neq z$. Chaque opération primitive modifie de façon particulière l'état de protection courant. Pour éclaircir parfaitement les choses, il est commode de considérer le concept de transition entre états. Soit θ une assignation et π une opération primitive. Supposons qu'il n'existe aucune variable dans $\theta(\pi)$, c'est-à-dire que chaque variable dans π est remplacée par une constante en utilisant θ. Si $\Delta = (S, O, A, B, P)$ et $\Delta' = (S', O', A', B', P')$ sont des états de protection alors nous dirons que Δ' est dérivable de Δ en utilisant θ et π, représenté par $\Delta \longrightarrow_{\pi}^{\theta} \Delta'$, ssi une des conditions suivantes est satisfaite :

– π est “créer le sujet x”, $\theta(x)$ n'appartient pas à S, $S' = S \cup \{\theta(x)\}$, $O' = O$, $A'[s, o] = A[s, o]$ pour tout $(s, o) \in S \times O$, $A'[\theta(x), o] = \emptyset$ pour tout $o \in O'$, $B'[s, o] = B[s, o]$ pour tout $(s, o) \in S \times O$, $B'[\theta(x), o] = \emptyset$ pour tout $o \in O'$, $P'[s, o] = P[s, o]$ pour tout $(s, o) \in S \times O$ et $P'[\theta(x), o] = \emptyset$ pour tout $o \in O'$;

– π est “détruire le sujet x”, $\theta(x)$ appartient à S, $S' = S \setminus \{\theta(x)\}$, $O' = O$, $A'[s, o] = A[s, o]$, $B'[s, o] = B[s, o]$ et $P'[s, o] = P[s, o]$ pour tout $(s, o) \in S' \times O'$;

– π est “créer l'objet y”, $\theta(y)$ n'appartient pas à O, $S' = S$, $O' = O \cup \{\theta(y)\}$, $A'[s, o] = A[s, o]$ pour tout $(s, o) \in S \times O$, $A'[s, \theta(y)] = \emptyset$ pour tout $s \in S'$, $B'[s, o] = B[s, o]$ pour tout $(s, o) \in S \times O$, $B'[s, \theta(y)] = \emptyset$ pour tout $s \in S'$, $P'[s, o] = P[s, o]$ pour tout $(s, o) \in S \times O$ et $P'[s, \theta(y)] = \emptyset$ pour tout $s \in S'$;

– π est “détruire l'objet y”, $\theta(y)$ appartient à O, $S' = S$, $O' = O \setminus \{\theta(y)\}$, $A'[s, o] = A[s, o]$, $B'[s, o] = B[s, o]$ et $P'[s, o] = P[s, o]$ pour tout $(s, o) \in S' \times O'$;

– π est "introduire r dans $A(x,y)$", $\theta(x)$ appartient à S, $\theta(y)$ appartient à O, $S' = S$, $O' = O$, pour tout $(s,o) \in S \times O$,

$$A'[s,o] = \begin{cases} A[s,o] \cup \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ A[s,o] & \text{sinon,} \end{cases}$$
$$B'[s,o] = \begin{cases} B[s,o] \cup \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ B[s,o] & \text{sinon,} \end{cases}$$
$$P'[s,o] = \begin{cases} P[s,o] \cup \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ P[s,o] & \text{sinon ;} \end{cases}$$

– π est "supprimer r de $A(x,y)$", $\theta(x)$ appartient à S, $\theta(y)$ appartient à O, $S' = S$, $O' = O$, pour tout $(s,o) \in S \times O$,

$$A'[s,o] = \begin{cases} A[s,o] \setminus \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ A[s,o] & \text{sinon,} \end{cases}$$

$B'[s,o] = B[s,o]$ et $P'[s,o] = P[s,o]$;

– π est "introduire r dans $B(x,y)$", $\theta(x)$ appartient à S, $\theta(y)$ appartient à O, $S' = S$, $O' = O$, pour tout $(s,o) \in S \times O$,

$A'[s,o] = A[s,o]$,

$$B'[s,o] = \begin{cases} B[s,o] \cup \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ B[s,o] & \text{sinon,} \end{cases}$$
$$P'[s,o] = \begin{cases} P[s,o] \cup \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ P[s,o] & \text{sinon ;} \end{cases}$$

– π est "supprimer r de $B(x,y)$", $\theta(x)$ appartient à S, $\theta(y)$ appartient à O, $S' = S$, $O' = O$, pour tout $(s,o) \in S \times O$,

$$A'[s,o] = \begin{cases} A[s,o] \setminus \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ A[s,o] & \text{sinon,} \end{cases}$$
$$B'[s,o] = \begin{cases} B[s,o] \setminus \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ B[s,o] & \text{sinon,} \end{cases}$$
$P'[s,o] = P[s,o]$;

– π est "introduire r dans $P(x,y)$", $\theta(x)$ appartient à S, $\theta(y)$ appartient à O, $S' = S$, $O' = O$, pour tout $(s,o) \in S \times O$,

$A'[s,o] = A[s,o]$, $B'[s,o] = B[s,o]$,

$$P'[s,o] = \begin{cases} P[s,o] \cup \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ P[s,o] & \text{sinon ;} \end{cases}$$

– π est "supprimer r de $P(x,y)$", $\theta(x)$ appartient à S, $\theta(y)$ appartient à O, $S' = S$, $O' = O$, pour tout $(s,o) \in S \times O$,

$$A'[s,o] = \begin{cases} A[s,o] \setminus \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ A[s,o] & \text{sinon,} \end{cases}$$
$$B'[s,o] = \begin{cases} B[s,o] \setminus \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ B[s,o] & \text{sinon,} \end{cases}$$
$$P'[s,o] = \begin{cases} P[s,o] \setminus \{r\} & \text{si } s = \theta(x) \text{ et } o = \theta(y) \\ P[s,o] & \text{sinon.} \end{cases}$$

3.5 Système de protection temporisée

On appellera condition atomique toute expression de la forme :

$$\zeta ::= r \in M[x,y]$$

où r est un droit appartenant à R, x une variable de type sujet, y une variable de type objet et M une des trois matrices A, B et P.

L'ensemble des conditions est engendré par la règle suivante :

$$C ::= \zeta \mid \neg C \mid C \vee C' \mid \forall x\, C \mid \forall y\, C$$

Ci-dessus, x et y sont respectivement une variable de type sujet et une variable de type objet. Soit C une condition, Δ un état de protection et θ une assignation. Supposons qu'il n'existe aucune variable libre dans $\theta(C)$, c'est-à-dire que toute variable appartenant à C est soit remplacée par une constante en utilisant θ ou bien elle est dans le champ d'un quantificateur. On dira que θ rend C vrai dans Δ, représenté par $\Delta \models_\theta C$ ssi :

$\Delta \models_\theta r \in M[x,y]$ ssi $\theta(x) \in S$, $\theta(y) \in O$ et $r \in M[\theta(x), \theta(y)]$

$\Delta \models_\theta \neg C$ ssi $\Delta \not\models_\theta C$

$\Delta \models_\theta C \vee C'$ ssi $\Delta \models_\theta C$ ou $\Delta \models_\theta C'$

$\Delta \models_\theta \forall x\ C$ ssi $\forall \theta' \approx_x \theta$ on a $\Delta \models_{\theta'} C$

$\Delta \models_\theta \forall y\ C$ ssi $\forall \theta' \approx_y \theta$ on a $\Delta \models_{\theta'} C$

Une condition temporisée Ct est de l'une des deux formes suivantes :

– "C depuis au moins la durée d" ;
– "C depuis au plus la durée d" ;

où d est un nombre réel positif.

Les opérations primitives peuvent être appliquées indirectement par l'intermédiaire de commandes temporisées de la forme :

– "si Ct_1 et ... et Ct_i alors début π_1 ; ... ; π_j fin" ;

où Ct_1, ..., Ct_i sont des conditions temporisées et π_1, ..., π_j sont des opérations primitives. Le nombre des conditions temporisées de la commande ci-dessus est l'entier non négatif i, et le nombre de ses opérations primitives est l'entier positif j. On trouvera des exemples de commades temporisées dans le tableau 3.1 page 36. Les commandes temporisées sont appliquées en remplaçant toutes les variables de la commande par des constantes. Ensuite, si les conditions initiales Ct_1, ..., Ct_i sont évaluées à vrai dans l'état de protection courant alors les opérations primitives π_1, ..., π_j sont éxécutées. Les commandes temporisées seront notées par les lettres α, α', etc, avec éventuellement des indices. On appelle système de protection temporisée, un ensemble fini $\{\alpha_1, \ldots, \alpha_k\}$ de commandes temporisées et un ensemble fini de droits R. Un système de protection temporisée est dit monotone ssi aucune de ses commandes temporisées ne contient une opération primitive de la forme "détruire" ou "supprimer". Il est monoconditionnel ssi aucune de ses commandes temporisées ne contient plus d'une condition temporisée tandis qu'il est mono-operationnel ssi aucune de ses commandes temporisées ne contient plus d'une opération primitive. Les systèmes de protection temporisée seront notés par les lettres Π, Π', etc, avec éventuellement des indices. Pour tout $i \in \{0, 1, 2, +\infty\}$ et pour tout $j \in \{1, 2, +\infty\}$, soit $\mathcal{C}(i, j)$ l'ensemble des systèmes de protection dont aucune des commandes temporisées ne contient plus de i conditions temporisées ou plus de j opérations primitives et $\mathcal{C}^+(i, j)$ l'ensemble de tous les systèmes de protection temporisée monotones appartenant à $\mathcal{C}(i, j)$.

3.6 Histoires

Une histoire est une application qui affecte à chaque nombre réel un état de protection. Les histoires seront notées par les lettres h, h', etc, avec éventuellement des indices. Nous nous intéressons aux histoires qui changent d'état un nombre fini de fois seulement dans tout intervalle fini. Nous supposerons donc que pour toute histoire h, il existe une séquence strictement croissante ..., v_{-1}, v_0, v_1, ... de nombres réels telle que $\lim_{n\to-\infty} v_n = -\infty$, $\lim_{n\to+\infty} v_n = +\infty$, et pour tout entier n, il existe un état de protection Δ_n tel que $h(v) = \Delta_n$ pour tout $v \in]v_n, v_{n+1}[$. Nous dirons que la séquence ..., (v_{-1}, Δ_{-1}), (v_0, Δ_0), (v_1, Δ_1), ... est une séquence discontinue pour h. Dans de telles séquences, les cinq composantes d'un état de protection Δ_n seront notées par S_n, O_n, A_n, B_n et P_n pour tout entier n. Soit θ une assignation et Ct une condition temporisée. Supposons qu'il n'existe aucune variable libre dans $\theta(Ct)$, c'est-à-dire que toute variable appartenant à Ct est soit remplacée par une constante en utilisant θ ou bien elle est dans le champ d'un quantificateur. Si h est une histoire avec une séquence discontinue ..., (v_{-1}, Δ_{-1}), (v_0, Δ_0), (v_1, Δ_1), ... et v un nombre réel alors on dira que θ rend Ct vrai dans h à v, représenté par $h, v \models_\theta Ct$, ssi l'une des deux conditions suivantes est satisfaite :

- Ct est "C depuis au moins la durée d" et pour tout entier n
  Si $v - d < v_{n+1}$ et $v_n < v$ alors $\Delta_n \models_\theta C$;

- Ct est "C depuis au plus la durée d" , et il existe un entier n tel que :
  $v - d < v_{n+1}$ et $v_n < v$ et $\Delta_n \not\models_\theta C$;

où d est un nombre réel non nul. La notion d'histoire est utilisée comme modèle pour le comportement des systèmes de protection temporisée. Nous dirons qu'une histoire h est un modèle pour un système de protection temporisée Π, noté $h \models \Pi$, ssi il existe une séquence discontinue ..., (v_{-1}, Δ_{-1}), (v_0, Δ_0), (v_1, Δ_1), ... pour h telle que pour tout entier n, si $\Delta_{n-1} \neq \Delta_n$ alors il existe une assignation θ et une commande temporisée $\alpha \in \Pi$ avec les conditions temporisées Ct_1, ..., Ct_i et les opérations primitives π_1, ..., π_j telles que $h, v_n \models_\theta Ct_1$, ..., $h, v_n \models_\theta Ct_i$ et $\Delta_{n-1} \longrightarrow^\theta_{\pi_1} \circ \ldots \circ \longrightarrow^\theta_{\pi_j} \Delta_n$.
Dans ce cas on peut représenter la séquence discontinue par :
..., $(v_{-1}, \Delta_{-1}) \longrightarrow^{\theta_0}_{\alpha_0} (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1)$, ... où $\alpha_0, \alpha_1 \ldots$ sont des commandes temporisées de Π et $\theta_0, \theta_1 \ldots$ des assignations.

3.7 Sécurité

On peut maintenant définir la notion de sûreté des systèmes de protection temporisée. Si h est une histoire de séquence discontinue ..., (v_{-1}, Δ_{-1}), (v_0, Δ_0), (v_1, Δ_1), ... et v un nombre réel alors nous dirons que h fait apparaître le droit r à v ssi il existe un sujet s, un objet o et un entier n tels que $v = v_n$, r appartient à $P_n[s, o]$ et r n'appartient pas à $P_{n-1}[s, o]$. Considérons un système de protection temporisée Π et un état de protection Δ. Si d est un nombre réel positif alors nous dirons que Π est d-attaqué pour le droit r par rapport à Δ ssi il existe une histoire h de séquence discontinue ..., (v_{-1}, Δ_{-1}), (v_0, Δ_0), (v_1, Δ_1), ... et un nombre réel v tels que $h \models \Pi$, h fait apparaître r à v, $v \leq v_0 + d$, et $\Delta_0 = \Delta$, $\Delta_{-1} = \Delta$, ... Nous dirons également que d est une période d'attente de Π pour r par rapport à Δ.

Soit $\mathcal{C}$ une classe de systèmes de protection temporisée. Les problèmes les plus simples qu'il nous est possible de poser à propos des systèmes de protection de $\mathcal{C}$ sont les deux problèmes de décision suivants :

Problème 1 : PROTECTION TEMPORISEE UNIVERSELLE($\mathcal{C}$),

Entrée : un système de protection temporisée $\Pi \in \mathcal{C}$, un droit r et un état de protection Δ,

Sortie : déterminer s'il existe un nombre réel positif d tel que Π est d-attaqué pour r par rapport à Δ.

Problème 2 : PROTECTION TEMPORISEE BORNEE($\mathcal{C}$),

Entrée : un système de protection temporisée $\Pi \in \mathcal{C}$, un droit r, un état de protection Δ, et un nombre réel positif d,

Sortie : déterminer si Π est d-attaqué pour r par rapport à Δ.

Supposons qu'il existe un nombre réel d tel que d est une période d'attente de Π pour le droit r par rapport à Δ. Par suite, il existe aussi une période d'attente de valeur minimale. Il se pourrait que d soit irrationnel. Puisque les contraintes temporisées dans Π sont rationnelles, la période d'attente de valeur minimale ne peut être que rationnelle. Ceci nous suggère de considérer le problème d'optimisation suivant :

Problème 3 : PROTECTION TEMPORISEE MINIMALE($\mathcal{C}$),

Entrée : un système de protection temporisée $\Pi \in \mathcal{C}$, un droit r et un état de protection Δ,

Sortie : trouver la plus petite période d'attente possible de Π pour r par rapport à Δ.

3.8 Disponibilité

Soit Π un système de protection temporisée et d un nombre réel positif non nul. On dit qu'un droit r est d-disponible pour le sujet s sur l'objet o relativement a l'état de protection $\Delta = (S, O, A, B, P)$ ssi pour tout histoire h, $h \models \Pi$, de séquence discontinue

$$\ldots, (v_{-1}, \Delta_{-1}) \longrightarrow^{\theta_0}_{\alpha_0} (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1), \ldots$$

telle que :

1. $\Delta_0 = \Delta$ et
2. $r \in B_0[s, o]$, $s \in S_0, o \in O_0$ et $r \notin B_{-1}[s, o]$, ou ($s \notin S_{-1}$ ou $o \notin O_{-1}$),

il existe un entier naturel $n \geq 0$ et une histoire h', $h' \models \Pi$, de séquence discontinue

$$\ldots, (v_{-1}, \Delta_{-1}) \longrightarrow^{\theta_0}_{\alpha_0} (v_0, \Delta_0) \longrightarrow^{\theta'_1}_{\alpha'_1} (v'_1, \Delta'_1), \ldots (v'_{n-1}, \Delta'_{n-1}) \longrightarrow^{\theta'_n}_{\alpha'_n}$$
$$(v'_n, \Delta'_n) \longrightarrow^{\theta'_{n+1}}_{\alpha'_{n+1}} (v'_{n+1}, \Delta'_{n+1}), \ldots$$

tels que : $v_n \leq v_0 + d$, $r \in A'_n[s, o]$, $s \in S_n$, $o \in O_n$ et pour tout entier naturel $n' \in \{0, ..., n\}$ on a $r \in B'_{n'}[s, o]$, $s \in S_{n'}$ et $o \in O_{n'}$.

Soit $\mathcal{C}$ une classe de systèmes de protection temporisée. Les problèmes les plus simples qu'il nous est possible de poser à propos de la disponibilité dans les systèmes de protection de $\mathcal{C}$ sont les deux problèmes de décision suivant :

Problème 1 : DISPONIBILITE UNIVERSELLE($\mathcal{C}$),

Entrée : un système de protection temporisée $\Pi \in \mathcal{C}$, un état de protection Δ, un droit r, un sujet $s \in S$ et un objet $o \in O$,

Sortie : déterminer s'il existe un nombre réel positif d tel que r est d-disponible pour s sur o relativement à Δ.

Problème 2 : PROTECTION TEMPORISEE BORNEE($\mathcal{C}$),

Entrée : un système de protection temporisée $\Pi \in \mathcal{C}$, un état de protection Δ, un droit r, un sujet $s \in S$, un objet $o \in O$ et un nombre réel positif d,

Sortie : déterminer si r est d-disponible pour s sur o relativement à Δ.

Supposons qu'il existe un nombre réel d tel que Π est d-disponible pour r. Par suite, il existe un plus petit nombre réel d' tel que Π est d'-disponible. Ceci nous suggère de considérer le problème d'optimisation suivant :

Problème 3 : DISPONIBILITE MINIMALE($\mathcal{C}$),

Entrée : un système de protection temporisée $\Pi \in \mathcal{C}$, un état de protection Δ, un droit r, un sujet $s \in S$ et un objet $o \in O$,

Sortie : trouver le plus petit nombre réel positif d pour lequel r est d-disponible pour s sur o relativement à Δ.

3.9 Exemple

Pour mieux comprendre la notion de disponibilité considérons l'exemple suivant. Soit Π un système de protection temporisée qui contient l'ensemble de droits $\{own, read, write\}$ noté $\{o, r, w\}$ et les treize commandes de la table 3.1. la commande (1) permet d'ajouter un nouvel utilisateur dans le réseau. Les commandes (2) et (3) permettent respectivement la création de nouveaux fichiers ainsi que l'introduction des droits de lecture et d'ecriture sur ces fichiers. La commande (4) permet de supprimer le droit d'écriture pour un certain sujet sur un certain fichier. Les commandes (5) et (6) permettent respectivement à un utilisateur de demander l'accès en lecture ou en écriture sur un fichier. Les commandes (7) et (8) permettent à un utilisateur d'arrêter la lecture respectivement l'écriture dans un fichier. Les commandes (9), (10), (11), (12) et (13) permettent sous certaines conditions d'accorder à l'utilisateur l'accès au fichier en lecture ou en écriture. Dans cet exemple, le même fichier peut être lu mais pas écrit par deux utilisateurs au même moment et deux utilisateurs, ne peuvent pas lire et écrire simultanément sur le même fichier .

Soit $\Delta = (S, O, A, B, P)$ l'état de protection représenté par les tables ci-dessous :

A	o_1
s_1	
s_2	w

B	o_1
s_1	r
s_2	w

P	o_1
s_1	r, w
s_2	o, r, w

Montrons que le droit r, est 6-disponible pour le sujet s_1 sur l'objet o_1 relativement à l'état de protection Δ.

1	"créer le sujet x ;"
2	"début créer l'objet y ; introduire o dans $P[x, y]$ fin"
3	"Si $o \in P[x, y]$ alors début introduire w dans $P[x', y]$; introduire r dans $P[x', y]$fin"
4	"Si $r \notin B[x, y]$ alors supprimer r de $P[x, y]$"
5	"Si $r \in P[x, y]$ alors introduire r dans $B[x, y]$"
6	"Si $w \in P[x, y]$ alors introduire w dans $B[x, y]$"
7	"Supprimer r de $B[x, y]$"
8	"Supprimer w de $B[x, y]$"
9	"Si $r \in B[x, y]$ et $\forall x\ w \notin A[x', y]$ alors introduire r dans $A[x, y]$ "
10	"Si $w \in B[x', y]$ et $\forall x\ w \notin A[x, y]$ et $\forall x\ r \notin A[x, y]$ alors introduire w dans $A[x', y]$ "
11	"Si $w \in B[x, y]$ depuis au moins la durée 4 et $w \in A[x', y]$ depuis au moins la durée 6 alors début supprimer w de $B[x', y]$; introduire w dans $A[x, y]$ fin"
12	"Si $r \in B[x, y]$ depuis au moins la durée 4 et $w \in A[x', y]$ depuis au moins la durée 6 alors début supprimer w de $B[x', y]$; introduire r dans $A[x, y]$ fin"
13	"Si $w \in B[x, y]$ depuis au moins la durée 4 et $r \in A[x', y]$ depuis au moins la durée 6 alors début supprimer r de $B[x', y]$; introduire w dans $A[x, y]$ fin"

TAB. 3.1 – Commandes du système de protection temporisée Π

Preuve:
Soit h une histoire, $h \models \Pi$, de séquence discontinue :

$$\ldots, (v_{-1}, \Delta_{-1}) \longrightarrow^{\theta_0}_{\alpha_0} (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1), \ldots (v_{n-1}, \Delta_{n-1}) \longrightarrow^{\theta_n}_{\alpha_n} (v_n, \Delta_n) \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} (v_{n+1}, \Delta_{n+1}), \ldots$$

On suppose que $\Delta_0 = \Delta$ et $r \notin B_{-1}[s_1, o_1]$ avec $s_1 \in S_{-1}$ et $o_1 \in O_{-1}$. D'après les commandes du système de protection temporisée Π, la commande α_0 ne peut être que la cinquième commande de la table 3.1 avec $\theta_0(x') = s_1$, $\theta_0(y) = o_1$. Dans l'état de protection Δ on a $w \in A[s_2, o_1]$. Il existe donc un entier $k < 0$ pour lequel la commande α_k permet d'introduire r dans $A[s_2, o_1]$ à l'instant v_k en utilisant la dixième, la onzième ou la treizième commande de la table 3.1. Dans ce cas v_k peut au plus être égale à v_{-1}.

Considérons l'histoire h', $h' \models \Pi$ de séquence discontinue :

$$\ldots, (v_{-1}, \Delta_{-1}) \longrightarrow^{\theta_0}_{\alpha_0} (v_0, \Delta_0) \longrightarrow^{\theta'_1}_{\alpha'_1} (v'_1, \Delta'_1), \ldots$$

où

- α'_1 est la douzième commande de la table 3.1 ;
- $\theta'_1(x') = s_1$, $\theta'_1(x') = s_2$, $\theta'_1(y) = o_1$;
- $\Delta'_1 = (S'_1, O'_1, A'_1, B'_1, P'_1)$ est l'état de protection représenté par les tables suivantes :

A'_1	o_1
s_1	r
s_2	w

B'_1	o_1
s_1	r
s_2	w

P'_1	o_1
s_1	r, w
s_2	o, r, w

- $\Delta'_2 = \Delta'_1$,$\Delta'_3 = \Delta'_1$,... ;

où v'_1 est égale au maximum entre $4 + v_0$ et $6 + v_k$. Nous savons que $6 + v_k \leq 6 + v_{-1}$. Donc v'_1 est inférieur à $v_0 + 6$.

On a ainsi pu trouver une histoire h' pour laquelle on a $v_1 \leq v_0 + 6$, $r \in A_1[s_1, o_1]$ et $r \in B_0[s, o]$. Le droit r est donc 6-disponible pour s_1 sur l'objet o_1 relativement à Δ. $\dashv$

Chapitre 4

Systèmes de protection temporisée spécifiques

4.1 Introduction

Nous avons présenté dans le chapitre précédent les systèmes de protection temporisée, trois problèmes de la protection ainsi que trois problèmes de la disponibilité. Il n'est pas facile de déterminer la décidabilité de ces six problèmes dans le cas général. C'est pourquoi nous allons nous restreindre dans ce chapitre aux états de protection temporisée qui ne contiennent qu'une seule matrice : la matrice des permissions. Egalement, nous allons nous restreindre à des systèmes de protection qui ont des commandes temporisées spécifiques [3]. Le problème de la disponibilité ne peut pas être défini dans ce contexte. Mais les résultats mathématiques de la décidabilité des trois problèmes de la protection temporisée pourront éventuellement être utilisés pour attaquer la question de la décidabilité du problème de la disponibilité.

4.2 Etats de protection

Comme pour le modèle HRU, un état de protection est un triplet (S, O, P) où S est un ensemble fini de sujets de cardinalité $\mid S \mid$, où O est un ensemble fini d'objets de cardinalité $\mid O \mid$ et où P est une matrice des permissions qui a une ligne pour chaque sujet de S et une colonne pour chaque objet de O. L'ensemble $P[s, o], s \in S, o \in O$, définit les droits que le sujet s a sur l'objet o. Les états de protection seront notés par les lettres Δ, Δ', etc, avec éventuellement des indices. La différence avec l'état de protection dans le modèle HRU, est que, comme au

P	o_1	o_2
s_1	o, r, w	r, w
s_2	r, w	
s_3		o

TAB. 4.1 – Etat de protection Δ.

chapitre 3, l'ensemble des sujets est disjoint de l'ensemble des objets. La table 4.1 illustre un état de protection simple Δ présenté sous la forme d'une matrice. Les éléments de la matrice spécifient les droits que chaque sujet a sur chaque objet. Les droits de la matrice sont les droits du système Unix : $own, read$ et $write$, notés o, r et w.

4.3 Transition entre états

Comme dans les deux chapitres précédents, les assignations remplacent les variables par des constantes. Elles seront aussi notées par les lettres θ, θ', etc, avec éventuellement des indices. Les opérations primitives sont identiques à celles du modèles HRU. La différence est que les modifications qu'apportent les deux opérations primitives "créer le sujet x" et "détruire le sujet x", lorsqu'elles sont appliquées à un état de protection, sont distinctes des modifications qu'elles apportent dans le modèle HRU. Cela est dû au fait de considérer que l'ensemble des sujets et l'ensemble des objets sont disjoints. Il n'est donc besoin que de préciser ici la sémantique des opérations primitives de création et de destruction des sujets. Soit θ une assignation, π_1 l'opération primitive "créer le sujet x", π_2 l'opération primitive "détruire le sujet x" et $\Delta = (S, O, P)$ et $\Delta' = (S', O', P')$ des états de protection. Supposons qu'il n'existe aucune variable dans $\theta(\pi_1)$ et dans $\theta(\pi_2)$.
Nous dirons que Δ' est dérivable de Δ en utilisant π_1 et θ, représenté par $\Delta \longrightarrow^{\theta}_{\pi_1} \Delta'$, ssi :

- $\theta(x)$ n'appartient pas à S, $S' = S \cup \{\theta(x)\}$, $O' = O$, $P'[s, o] = P[s, o]$ pour tout $(s, o) \in S \times O$ et $P'[\theta(x), o] = \emptyset$ pour tout $o \in O'$;

et que Δ' est dérivable de Δ en utilisant π_2 et θ, représenté par $\Delta \longrightarrow^{\theta}_{\pi_1} \Delta'$, ssi :

- $\theta(x)$ appartient à S, $S' = S \setminus \{\theta(x)\}$, $O' = O$ et $P'[s, o] = P[s, o]$ pour tout $(s, o) \in S' \times O'$.

4.4 Système de protection temporisée spécifique

Les opérations primitives peuvent être appliquées indirectement par l'intermédiaire de commandes temporisées spécifiques de la forme :

– "si Ct_1 et ... et Ct_i alors début π_1 ; ... ; π_j fin",

où Ct_1, ..., Ct_i sont des conditions temporisées de la forme :

– "$r \in P[x, y]$ depuis au moins la durée d" où d est un nombre réel positif,

et π_1, ..., π_j sont des opérations primitives. Le nombre des conditions temporisées de cette commande est l'entier non négatif i, et le nombre de ses opérations primitives est l'entier positif j. Les commandes temporisées spécifiques sont appliquées en remplaçant toutes les variables de la commande par des constantes. Ensuite, si les conditions initiales Ct_1, ..., Ct_i sont évaluées à vrai dans l'état de protection courant alors les opérations primitives π_1, ..., π_j sont exécutées. Les commandes temporisées spécifiques seront notées par les lettres α, α', etc, avec éventuellement des indices.

On appelle système de protection temporisée spécifique, un ensemble fini $\{\alpha_1, \ldots, \alpha_k\}$ de commandes temporisées spécifiques et un ensemble fini de droits R. Comme pour les systèmes de protection temporisée, un système de protection temporisée spécifique est dit monotone ssi aucune de ses commandes temporisées spécifiques ne contient une opération primitive de la forme "détruire" ou "supprimer". Il est monoconditionnel ssi aucune de ses commandes temporisées spécifiques ne contient plus d'une condition temporisée, tandis qu'il est monooperationnel ssi aucune de ses commandes temporisées spécifiques ne contient plus d'une opération primitive. Les systèmes de protection temporisée spécifiques seront notés par les lettres Π, Π', etc, avec éventuellement des indices. Pour tout $i \in \{0, 1, 2, +\infty\}$ et pour tout $j \in \{1, 2, +\infty\}$, soit $\mathcal{C}(i, j)$ l'ensemble des systèmes de protection temporisée spécifiques dont aucune des commandes temporisées ne contient plus de i conditions temporisées ou plus de j opérations primitives et soit $\mathcal{C}^+(i, j)$ l'ensemble de tous les systèmes de protection temporisée spécifiques monotones appartenant à $\mathcal{C}(i, j)$.

4.5 Exemple

Cet exemple concerne la gestion d'un entrepôt de produit. Pour chaque produit stocké on a un fichier qui contient toutes les informations le concernant. Ce fichier peut être consulté pour être modifié ou pour être lu. Une fois le produit vendu, on supprime son fichier. Le but du gérant est de vendre en priorité les produits stockés

depuis le plus longtemps, c'est-à-dire depuis au moins une durée qu'il fixera par exemple à 3 semaines. Dans cette exemple l'ensemble des sujets sera l'ensemble du personnel, l'ensemble des objets sera l'ensemble des fichiers associés aux produits et l'ensemble des droits est l'ensemble $R = \{r, w, o\}$. Un système de protection temporisée spécifique Π pourra être défini par l'ensemble R et les commandes ci-dessous.

1	"créer le sujet x"
2	"début créer l'objet y ; introduire o dans $P[x, y]$ fin"
3	"Si $o \in P[x, y]$ alors début introduire w dans $P[x', y]$; introduire r dans $P[x', y]$ fin"
4	"Supprimer r de $P[x, y]$"
5	"Supprimer w de $P[x, y]$"
6	"Si $o \in P[x, y]$ depuis au moins la durée 3 alors détruire l'objet y"

4.6 Sécurité

Dans le cadre des systèmes de protection temporisée spécifiques, les notions d'histoire et d'apparition des droits sont identiques à celles introduites dans le chapitre précédent. Même chose pour la notion de systèmes de protection temporisée spécifiques d-attaqués.

Soit Π un système de protection temporisée spécifique. Si Π' est un système de protection temporisée spécifique obtenu à partir de Π en modifiant toutes ou une partie seulement des durée de ses contraintes temporisées "depuis au moins la durée d" alors il est évident que Π et Π' font apparaître les mêmes droits, mais pas forcément au même moment. Soit $HRU(\Pi)$ le système de protection HRU obtenu à partir de Π en éliminant la partie "depuis au moins la durée d" de toutes les contraintes temporisées. On a donc :

Lemme 1 *Soit Π un système de protection temporisée spécifique, r un droit et Δ un état de protection. Les conditions suivantes sont équivalentes :*

- *Il existe un nombre réel positif d, tel que Π est d-attaqué pour r par rapport à Δ.*
- *$HRU(\Pi)$ est non sûr pour r relativement à Δ.*

Nous avons décrit dans le chapitre précédent, trois problèmes de la protection : UNIVERSELLE($\mathcal{C}$), BORNÉE($\mathcal{C}$) et MINIMALE($\mathcal{C}$) pour des classes quelquonques de systèmes de protection temporisée. Nous n'avons donné aucun résultat mathématique concernant la décidabilité de ces problèmes. Dans le cadre

restreint de sytèmes de protection temporisée spécifiques nous avons considéré le problème de décision suivant :

Problème 1 : PROTECTION TEMPORISEE UNIVERSELLE($\mathcal{C}$),

Entrée : un système de protection temporisée spécifique $\Pi \in \mathcal{C}$, un droit r et un état de protection Δ,

Sortie : déterminer s'il existe un nombre réel positif d tel que Π est d-attaqué pour r par rapport à Δ.

Comme tout problème de décision, PROTECTION TEMPORISEE UNIVERSELLE($\mathcal{C}$) doit avoir un ensemble dénombrable d'instances. Pour cette raison, à partir de maintenant, nous supposerons que pour tout nombre réel positif d, si un système de protection temporisée spécifique $\Pi \in \mathcal{C}$ contient la contrainte temporisée "depuis au moins la durée d" alors d est rationnel.

Théorème 6

- *PROTECTION TEMPORISEE UNIVERSELLE*($\mathcal{C}(+\infty, +\infty)$) *est indécidable,*
- *PROTECTION TEMPORISEE UNIVERSELLE*($\mathcal{C}^+(+\infty, +\infty)$) *est indécidable,*
- *PROTECTION TEMPORISEE UNIVERSELLE*($\mathcal{C}(+\infty, 1)$) *est décidable,*
- *PROTECTION TEMPORISEE UNIVERSELLE*($\mathcal{C}^+(1, +\infty)$) *est décidable.*

Preuve: Des résultats correspondants pour les systèmes de protection HRU ont été obtenus par [13, 12, 14]. Le lemme 1 nous permet de conclure. $\dashv$

Le deuxième problème posé à propos des systèmes de protection temporisée de $\mathcal{C}$ est le problème de décision suivant :

Problème 2 : PROTECTION TEMPORISEE BORNEE($\mathcal{C}$),

Entrée : un système de protection temporisée spécifique $\Pi \in \mathcal{C}$, un droit r, un état de protection Δ, et un nombre réel positif d,

Sortie : déterminer si Π est d-attaqué pour r par rapport à Δ.

Le résultat suivant est une conséquence immédiate des résultats établis dans le théorème 8 et le théorème 9.

Théorème 7

- *PROTECTION TEMPORISEE BORNEE*$(\mathcal{C}(+\infty, 1))$ *est décidable,*
- *PROTECTION TEMPORISEE BORNEE*$(\mathcal{C}^+(1, +\infty))$ *est décidable.*

Nous ignorons si PROTECTION TEMPORISEE BORNEE$(\mathcal{C}(+\infty, +\infty))$ et PROTECTION TEMPORISEE BORNEE$(\mathcal{C}^+(+\infty, +\infty))$ sont indécidables.

Considérons maintenant le troisième problème, qui est un problème d'optimisation.

Problème 3 : PROTECTION TEMPORISEE MINIMALE$(\mathcal{C})$,

Entrée : un système de protection temporisée spécifique $\Pi \in \mathcal{C}$, un droit r et un état de protection Δ,

Sortie : trouver la plus petite période d'attente possible de Π pour r par rapport à Δ.

Si nous disposions d'un algorithme A permettant de résoudre PROTECTION TEMPORISEE MINIMALE$(\mathcal{C})$ alors nous serions capable de construire un algorithme permettant de décider PROTECTION TEMPORISEE UNIVERSELLE$(\mathcal{C})$: étant donnée une entrée (r, Π, Δ), nous serions capable de décider s'il existe un nombre réel positif d tel que Π est d-attaqué pour r par rapport à Δ en demandant simplement si $A(r, \Pi, \Delta)$ est défini. Par conséquent, PROTECTION TEMPORISEE MINIMALE$(\mathcal{C})$ est un problème au moins aussi difficile que PROTECTION TEMPORISEE UNIVERSELLE$(\mathcal{C})$. Pour de semblables raisons, il est facile de voir que PROTECTION TEMPORISEE MINIMALE$(\mathcal{C})$ est un problème au moins aussi difficile que PROTECTION TEMPORISEE BORNEE$(\mathcal{C})$. Il s'ensuit que : si PROTECTION TEMPORISEE UNIVERSELLE est indécidable pour $\mathcal{C}$ ou si PROTECTION TEMPORISEE BORNEE est indécidable pour $\mathcal{C}$, alors PROTECTION TEMPORISEE MINIMALE l'est aussi. Nous pouvons donc dire que :

- PROTECTION TEMPORISEE MINIMALE$(\mathcal{C}(+\infty, +\infty))$ et PROTECTION TEMPORISEE MINIMALE$(\mathcal{C}^+(+\infty, +\infty))$ sont indécidables.

Il s'ensuit aussi que : si PROTECTION TEMPORISEE MINIMALE est décidable pour $\mathcal{C}$ alors PROTECTION TEMPORISEE BORNEE est décidable pour $\mathcal{C}$. Nous pouvons maintenant donner la preuve du théorème suivant.

Théorème 8

– *PROTECTION TEMPORISEE MINIMALE*$(\mathcal{C}(+\infty, 1))$
est décidable.

Preuve:

Remarque

1. Sans perte de généralité, nous supposons que lorsqu'un sujet ou un objet est créé dans une séquence discontinue, $\ldots (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{m-1}}_{\alpha_{m-1}} (v_{m-1}, \Delta_{m-1}) \longrightarrow^{\theta_m}_{\alpha_m} (v_m, \Delta_m), \ldots$, c'est qu'il n'apparaît pas avant dans la séquence,
2. Dans un système de protection temporisée spécifique mono-opérationnel, la forme d'une commande est égale à la forme de l'unique opération primitive qu'elle contient.

Lemme 2 *Soit* Π *un système de protection temporisée spécifique mono-opérationnel,* r *un droit,* v *un nombre réel positif,* Δ *un état de protection et* h *une histoire,* $h \models \Pi$*, de séquence discontinue :*

$$\ldots (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{m-1}}_{\alpha_{m-1}} (v_{m-1}, \Delta_{m-1}) \longrightarrow^{\theta_m}_{\alpha_m} (v_m, \Delta_m) \ldots (1)$$

où $\Delta_0 = \Delta$, $\Delta_{-1} = \Delta$, *... et où* $v_m = v$*. Supposons que* h *fait apparaître* r *à* v_m *en utilisant un nombre minimum de commandes. Pour tout entier* $l \in \{1, \ldots, m-1\}$*, aucune commande* α_l *n'est de la forme "détruire le sujet* x*", ou de la forme "détruire l'objet* y*".*

Preuve:

Soit α_n la dernière commande de la forme "détruire le sujet x" telle que $\theta_n(x) = s_n$ (respectivement "détruire l'objet y" avec $\theta_n(y) = o_n$).
Considérons l'histoire h' de séquence discontinue :

$$\ldots, (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{n-1}}_{\alpha_{n-1}} (v_{n-1}, \Delta_{n-1}) \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} (v_{n+1}, \Delta'_{n+1})$$

$$\longrightarrow^{\theta'_{n+2}}_{\alpha_{n+2}} \ldots \longrightarrow^{\theta'_m}_{\alpha_m} (v_m, \Delta'_m), \ldots (2)$$

telle que pour tout entier $l \in \{n+1, \ldots, m\}$:
– $S'_l = S_l \cup \{s_n\}$ (resp $S'_l = S_l$) ;
– $O'_l = O_l$ (resp $O'_l = O_l \cup \{o_n\}$) ;

– $\theta'_l(x) = \theta_l(x)$ pour toute variable x de type sujet ;
– $\theta'_l(y) = \theta_l(y)$ pour toute variable y de type objet et

– $P'_l[s,o] = \begin{cases} P_{n-1}[s,o] & \text{si } s = s_n, s \in S_l \text{ (resp } o = o_n, o \in O_l) \\ P_l[s,o] & \text{sinon.} \end{cases}$

Le lecteur vérifiera que les assignations et les commandes permettent de passer de Δ_0 à Δ'_m. La raison est que les commandes ne testent pas l'absence d'un droit dans une cellule (on a pas de commande avec conditions négatives) et que $P_l[s,o] \subseteq P'_l[s,o]$, pour tout entier $l \in \{0,\ldots,n-1,n+1,\ldots,m\}$, pour tout sujet $s \in S_l$ et pour tout objet $o \in O_l$. De plus les commandes sont éxecutées au même moment que dans la séquence initiale. Si le droit r apparaît dans Δ_m à v_m alors il apparaîtra dans Δ'_m à v_m. Nous avons donc pu trouver une histoire qui fait apparaître r à v_m en utilisant une commande de moins que la séquence utilisant un nombre minimum de commandes, ce qui est absurde. On ne peut pas, de ce fait, utiliser une commande de la forme "détruire le sujet x" ou de la forme "détruire l'objet y" pour la construction d'une séquence minimale d'états de protection, qui fait apparaître r à v_m. ⊣

Lemme 3 *Soit Π un système de protection temporisée spécifique mono-opérationnel, r un droit, v un nombre réel positif, Δ un état de protection et h une histoire, $h \models \Pi$, de séquence discontinue :*

$$\ldots (v_0,\Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1,\Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{m-1}}_{\alpha_{m-1}} (v_{m-1},\Delta_{m-1}) \longrightarrow^{\theta_m}_{\alpha_m} (v_m,\Delta_m)\ldots(1)$$

où $\Delta_0 = \Delta$, $\Delta_{-1} = \Delta$, ... et où $v_m = v$. Supposons que h fait apparaître r à v_m en utilisant un nombre minimum de commandes. Il existe au plus une commande $\alpha_l, l \in \{1,\ldots,m-1\}$ de la forme "supprimer r_l de $P[x,y]$".

Preuve:

1. Il n'existe aucune commande de la forme "supprimer r' de $P[x,y]$" ($r' \neq r$).

   Preuve: Soit α_n la dernière commande de la forme "supprimer r_n de $P[x,y]$" ($r_n \neq r$) telle que $\theta_n(x) = s_n$ et $\theta_n(y) = o_n$.
   Considérons la séquence suivante :

$$\ldots, (v_0,\Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1,\Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{n-1}}_{\alpha_{n-1}} (v_{n-1},\Delta_{n-1}) \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} (v_{n+1},\Delta'_{n+1})$$

$$\longrightarrow_{\alpha_{n+2}}^{\theta'_{n+2}} \ldots \longrightarrow_{\alpha_m}^{\theta'_m} (v_m, \Delta'_m), \ldots (3)$$

telle que pour tout entier $l \in \{n+1, \ldots, m\}$:
– $S'_l = S_l$,
– $O'_l = O_l$,
– $\theta'_l(x) = \theta_l(x)$ pour toute variable x de type sujet,
– $\theta'_l(y) = \theta_l(y)$ pour toute variable y de type objet et

– $P'_l[s,o] = \begin{cases} P_l[s,o] \cup \{r'\} & \text{si } s = s_n \text{ et } o = o_n \\ P_l[s,o] & \text{sinon.} \end{cases}$

Il est facile de vérifier que les commandes de cette séquence, en utilisant les assignations associées, permettent de passer de Δ_0 à Δ'_m. En effet, $P_l[s,o] \subseteq P'_l[s,o]$ pour tout entier $l \in \{0, \ldots, n-1, n+1, \ldots, m\}$, pour tout sujet $s \in S_l$ et pour tout objet $o \in O_l$. Le fait d'exécuter les commandes au même moment dans la séquence (1) et dans la séquence (3) est important pour que les commandes soient applicables. Si le droit r apparaît dans Δ_m à v_m alors il apparaîtra dans Δ'_m à v_m, ce qui est absurde. Nous avons pu construire une séquence de longueur inférieure à celle de la séquence qui utilise un nombre de commandes minimum pour faire apparaître r à v_m. Donc on ne peut pas avoir une commande de la forme "supprimer r' de $P[x,y]$" ($r' \neq r$) dans la séquence minimale. ⊣

2. Il existe au plus une commande de la forme "supprimer r de $P[x,y]$".

Preuve: Soit α_{n-q} et α_n deux commandes de la forme "supprimer r de $P[x,y]$" telles que :
– $\theta_{n-q}(x) = s_{n-q}$,
– $\theta_{n-q}(y) = o_{n-q}$,
– $\theta_n(x) = s_n$ et
– $\theta_n(y) = o_n$.

Remarque
Les cellules (s_{n-q}, o_{n-q}) et (s_n, o_n) sont distinctes. En effet, leur égalité contredirait la minimalité de la séquence.

Il existe deux possibilités pour l'apparition du droit r. Il peut apparaître dans une des deux cellules (s_{n-q}, o_{n-q}), (s_n, o_n) ou dans une cellule (s,o) différente des deux cellules précédentes.

(a) Le droit r apparaît dans la cellule (s_n, o_n) ou dans la cellule (s_{n-q}, o_{n-q}) :

Nous supposons que r apparaît dans la cellule (s_n, o_n) (nous avons le même argument si r apparaît dans la cellule (s_{n-q}, o_{n-q})). Considérons la séquence

$$\ldots, (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{n-q-1}}_{\alpha_{n-q-1}} (v_{n-q-1}, \Delta_{n-q-1}) \longrightarrow^{\theta'_{n-q+1}}_{\alpha_{n-q+1}} (v_{n-q+1}, \Delta'_{n-q+1}) \longrightarrow^{\theta'_{n-q+2}}_{\alpha_{n-q+1}} \ldots \longrightarrow^{\theta'_m}_{\alpha_m} (v_m, \Delta'_m), \ldots (4)$$

telle que pour tout entier $l \in \{n-q+1, \ldots, m\}$:
– $S'_l = S_l$,
– $O'_l = O_l$;
– $\theta'_l(x) = \theta_l(x)$ pour toute variable x de type sujet ;
– $\theta'_l(y) = \theta_l(y)$ pour toute variable y de type objet et

– $P'_l[s,o] = \begin{cases} P_l[s,o] \cup \{r\} & \text{si } s = s_{n-q} \text{ et } o = o_{n-q} \\ P_l[s,o] & \text{sinon.} \end{cases}$

Cette séquence a une commande de moins (α_{n-q}) que la séquence (1). Il est facile de vérifier que les commandes de cette séquence permettent de passer de Δ_0 à Δ'_m en utilisant les assignations associées. En effet, $P_l[s,o] \subseteq P'_l[s,o]$ pour tout entier $l \in \{0, \ldots, n-q-1, n-q+1, \ldots, m\}$, pour tout sujet $s \in S_l$ et pour tout objet $o \in O_l$. Si le droit r apparaît dans Δ_m à v_m alors il apparaîtra dans Δ'_m à v_m. Nous avons donc pu construire une séquence qui fait apparaître r à v_m, plus courte que la séquence de longueur minimale, ce qui est absurde. Il n'est donc pas possible d'avoir dans la séquence minimale plus d'une commande de la forme "supprimer r de $P[x,y]$".

(b) Le droit r apparaît dans une cellule (s,o) différentes des cellules, (s_{n-q}, o_{n-q}) et (s_n, o_n) :
Considérons la séquence

$$\ldots, (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} (v_2, \Delta_2) \ldots \longrightarrow^{\theta_{n-q-1}}_{\alpha_{n-q-1}} (v_{n-q-1}, \Delta_{n-q-1}) \longrightarrow^{\theta'_{n-q+1}}_{\alpha_{n-q+1}} (v_{n-q+1}, \Delta'_{n-q+1}) \longrightarrow^{\theta'_{n-q+2}}_{\alpha_{n-q+2}} (v_{n-q+2}, \Delta'_{n-q+2}) \ldots \longrightarrow^{\theta'_{n-1}}_{\alpha_{n-1}} (v_{n-1}, \Delta'_{n-1}) \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} (v_{n+1}, \Delta'_{n+1})$$

$$\longrightarrow^{\theta'_{n+2}}_{\alpha_{n+2}} (v_{n+2}, \Delta'_{n+2}) \ldots \longrightarrow^{\theta'_m}_{\alpha_m} (v_m, \Delta'_m) \ldots (5)$$

telle que pour tout entier $l \in \{n-q+1, \ldots, m\}$:
- $S'_l = S_l$;
- $O'_l = O_l$;
- $\theta'_l(x) = \theta_l(x)$ pour toute variable x de type sujet ;
- $\theta'_l(y) = \theta_l(y)$ pour toute variable y de type objet et

- $P'_l[s,o] = \begin{cases} P_l[s,o] \cup \{r\} & \text{si } (s,o) = (s_{n-q}, o_{n-q}) \\ & \text{ou } (s,o) = (s_n, o_n) \\ P_l[s,o] & \text{sinon.} \end{cases}$

Cette séquence a les commandes α_{n-q} et α_n de moins que la séquence (1). Les commandes α_l, $l > n-q$ restent applicables aux instants v_l en utilisant les assignations associées. En effet, les commandes ne testent pas l'absence d'un droit dans une cellule. Il est facile de vérifier que si le droit r apparaît dans Δ_m à v_m alors il apparaîtra dans Δ'_m à v_m. Nous avons donc montré dans les deux cas possibles pour l'apparition du droit r qu'on ne pouvait pas avoir dans la séquence minimal plus d'une commande de la forme "supprimer r de $P[x,y]$". Ceci termine la preuve de la deuxième partie du lemme 3.

⊣

Le fait d'avoir au plus une commande de la forme "supprimer", dans la séquence minimale est donc maintenant démontré.

⊣

Lemme 4 *Soit Π un système de protection temporisée spécifique mono-opérationnel, r un droit, v un nombre réel positif, Δ un état de protection et h une histoire, $h \models \Pi$, de séquence discontinue :*

$$\ldots (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{m-1}}_{\alpha_{m-1}} (v_{m-1}, \Delta_{m-1}) \longrightarrow^{\theta_m}_{\alpha_m} (v_m, \Delta_m) \ldots, (1)$$

où $\Delta_0 = \Delta$, $\Delta_{-1} = \Delta$, ... et où $v_m = v$. Supposons que h fait apparaître r à v_m en utilisant un nombre minimum de commandes. Il existe au plus une commande $\alpha_l, l \in \{1, \ldots, m-1\}$ de la forme "créer le sujet x " et il existe au plus une commande $\alpha_q, q \in \{1, \ldots, m-1\}$ de la forme "créer l'objet y ".

Preuve:

Supposons qu'il existe deux commandes α_l et α_q de la forme "céer le sujet x" avec l'assignation $\theta_l(x) = s_l$ et $\theta_q(x) = s_q, l < q$ où $l, q \in \{1, \ldots, m-1\}$ (de la même façon nous pouvons considérer que les deux commandes sont de la forme "créer l'objet y"). Soit la séquence obtenue en supprimant la commande α_q :

$$\ldots (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{l-1}}_{\alpha_{l-1}} (v_{l-1}, \Delta_{l-1}) \longrightarrow^{\theta_l}_{\alpha_l}$$
$$(v_l, \Delta_l) \longrightarrow^{\theta_{l+1}}_{\alpha_{l+1}} \ldots \longrightarrow^{\theta_{q-1}}_{\alpha_{q-1}} (v_{q-1}, \Delta_{q-1})$$

$$\longrightarrow^{\theta'_{q+1}}_{\alpha_{q+1}} (v_{q+1}, \Delta'_{q+1}) \longrightarrow^{\theta'_{q+2}}_{\alpha_{q+2}} (v_{q+2}, \Delta'_{q+2}) \ldots \longrightarrow^{\theta'_m}_{\alpha_m} (v_m, \Delta'_m) \ldots (6)$$

telle que pour tout entier $t \in \{q+1, \ldots, m\}$:
- $S'_t = S_t \setminus \{s_q\}$;
- $O'_t = O_t$;
- $\theta'_t(x) = \begin{cases} s_l & \text{si } \theta_t(x) = s_q \\ \theta_t(x) & \text{sinon} ; \end{cases}$
- $\theta'_t(y) = \theta_t(y)$ pour toute variable y de type objet et
- $P'_t[s, o] = \begin{cases} P_t[s, o] \cup P_t[s_q, o] & \text{si } s = s_l \\ P_t[s, o] & \text{sinon.} \end{cases}$

Il est simple de vérifier qu'en utilisant les assignations associées, les commandes de cette séquence permettent de passer de Δ_0 à Δ'_m. Si le droit r apparaît dans $(s_q, o), o \in O_m$, à v_m dans la séquence minimale alors l'apparition de r se fera dans (s_l, o) à v_m dans la séquence (6). Sinon l'apparition de r se fera dans la même cellule dans Δ'_m et Δ_m à v_m. Nous avons pu construire une séquence de longueur inférieure à celle de la séquence minimale, nous avons donc une contradiction. De ce fait on ne peut pas avoir plus d'une commande de la forme créer "créer le sujet x" ou "créer l'objet y" dans la séquence de longueur minimale qui fait apparaître r à v. $\dashv$

Les lemmes 2, 3 et 4 nous permettent de conclure que la séquence discontinue qui permet de faire apparaître le droit r à un instant v donné, en utilisant un nombre minimum de commandes, a les propriétés suivantes :
- le nombre de commandes de la forme "créer le sujet x" est au plus égal à 1 ;
- le nombre de commandes de la forme "créer l'objet y" est au plus égal à 1 ;
- le nombre de commandes de la forme "supprimer r_l de $P[x, y]$", $l \in \{1, \ldots, m-1\}$ est au plus égal à 1 et
- le nombre de commandes de la forme "ajouter r_l dans $P[x, y]$", $r_l \neq r, l \in \{1, \ldots, m-1\}$ est au plus égal à $(|S_0|+1) \times (|O_0|+1) \times |R-1|$.

Le nombre de commandes de la séquence minimale qui permet d'arriver à l'apparition à un instant quelconque v est donc borné par $b = (|S_0|+1) \times (|O_0|+1) \times |R-1| + 4$ (la commande α_m est comptée). Cette borne ne dépend pas de

v. Le fait de borner le nombre de commandes nous permet de construire un algorithme qui testera toutes les combinaisons possibles d'au plus b commandes. Cela nous permet également de trouver l'histoire qui fera apparaître r le plus tôt. Nous pourrons ainsi déterminer la valeur du plus petit nombre réel positif d tel que Π est d-attaqué.
$\dashv$

Nous avons ainsi montrer que le problème de la PROTECTION TEMPORISEE MINIMALE($\mathcal{C}(+\infty, 1)$) est décidable. Nous abordons maintenant la décidabilité du problème de la protection temporisée minimale pour la classe des systèmes de protection temporisée spécifiques monotones monoconditionnels.

Théorème 9 – *PROTECTION TEMPORISEE MINIMALE*($\mathcal{C}^+(1, +\infty)$) *est décidable.*

Preuve:

Pour démontrer la décidabilité de notre problème de décision, le consept de chaîne est essentielle.

Définition 1 *Soit* Π *un système de protection temporisée spécifique monotone monoconditionnel,* Δ *un état de protection et* h *une histoire,* $h \models \Pi$*, de séquence discontinue :*

$$\ldots, (\Delta_0, v_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_n}_{\alpha_n} (v_n, \Delta_n) \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} (v_{n+1}, \Delta_{n+1}), \ldots$$

où $\Delta_0 = \Delta$, $\Delta_{-1} = \Delta$, ...
On dira que la séquence ci-dessus est une chaîne de $r_1 \in (s_1, o_1)$, $s_1 \in S_1$, $o_1 \in O_1$ *à* $r_{n+1} \in (s_{n+1}, o_{n+1})$, $s_{n+1} \in S_{n+1}$, $o_{n+1} \in O_{n+1}$ *lorsque la condition suivante est vérifiée pour tout entier* $l \in \{1, \ldots, n+1\}$ *:*

- *si "*$r_l \in P[x, y]$*" est la condition de la commande* α_l *alors on a* $r_l \in P_{l-1}[\theta_l(x), \theta_l(y)]$ *et (*$r_l \notin P_{l-2}[\theta_l(x), \theta_l(y)]$ *ou* $\theta_l(x) \notin S_{l-2}$ *ou* $\theta_l(y) \notin O_{l-2}$*) avec* $\theta_l(x) = s_l$ *et* $\theta_l(y) = o_l$*.*

Remarque :

1. Sans perte de généralité, nous supposons que $|S_0| \neq 0$ et $|O_0| \neq 0$.
2. Si $r' \notin P_l[s, o]$ alors pour tout entier $q \in \{0, ..., l-1\}$, si $s \in S_q$ et $o \in O_q$ alors $r' \notin P_q[s, o]$ si $s \in S_q$ et $o \in O_q$ pour tout entier $0 \leqslant q \leqslant l-1$, car le système de protection temporisée spécifique Π considéré est monotone.

Considérons d'abord le cas où aucune commande de la séquence qui fait apparaître r n'a sa condition qui porte sur une nouvelle cellule.

Lemme 5 *Soit Π un système de protection temporisée, spécifique, monotone et monoconditionnel, r un droit, v un nombre réel positif, Δ un état de protection et h une histoire, $h \models \Pi$, de séquence discontinue :*

$$\ldots (v_0, \Delta_0) \longrightarrow_{\alpha_1}^{\theta_1} (v_1, \Delta_1) \longrightarrow_{\alpha_2}^{\theta_2} \ldots \longrightarrow_{\alpha_n}^{\theta_n} (v_n, \Delta_n) \longrightarrow_{\alpha_{n+1}}^{\theta_{n+1}} (v_{n+1}, \Delta_{n+1}) \ldots, (1)$$

où $\Delta_0 = \Delta$, $\Delta_{-1} = \Delta$, ... et où $v_{n+1} = v$. Supposons que h fait apparaître r à v_{n+1} en utilisant un nombre minimum de commandes. Si toute commande $\alpha_l, 1 \leq l \leq n$, est telle que sa condition "si $r_l \in P[x, y]$" vérifie que :

- *$\theta_l(x) \in S_0$ et*
- *$\theta_l(y) \in O_0$*

alors $n \leq |S_0| \times |O_0| \times (|R| - 1) + 2$.

Preuve:

Soit (s_*, o_*) la cellule dans laquelle le droit r apparaît. On distingue deux cas :

1. $s_* \in S_0$ et $o_* \in O_0$,
2. $s_* \notin S_0$ ou $s_* \notin O_0$.

1. $s_* \in S_0$ et $o_* \in O_0$
   Dans ce cas on a les deux propriétés (a) et (b) suivantes :
   (a) Pour tout entier $l \in \{1, \ldots, n\}$, le corps de la commande α_l contient une opération primitive de la forme "introduire r' dans $P[x, y]$", $r' \neq r$:

   Preuve: Soit α_q, une commande qui ne contient aucune opération primitive de la forme "introduire r' dans $P[x, y]$". Π est un système de protection temporisée spécifique monotone monoconditionnel, le corps de la commande α_q ne peut donc contenir que des opérations primitives de la forme "créer le sujet x" ou de la forme "créer l'objet y". Soit $s_1, \ldots, s_m$ et $o_1, \ldots, o_{m'}$ les sujets et les objets créés par la commande α_q. Considérons la séquence discontinue :

$$\ldots, (v_0, \Delta_0) \longrightarrow_{\alpha_1}^{\theta_1} (v_1, \Delta_1) \longrightarrow_{\alpha_2}^{\theta_2} \ldots \longrightarrow_{\alpha_{q-1}}^{\theta_{q-1}} (v_{q-1}, \Delta_{q-1}) \longrightarrow_{\alpha_{q+1}}^{\theta'_{q+1}} (v_{q+1}, \Delta'_{q+1}) \longrightarrow_{\alpha_{q+2}}^{\theta'_{q+2}} (v_{q+2}, \Delta'_{q+2}) \ldots \longrightarrow_{\alpha_{n+1}}^{\theta'_{n+1}} (v_{n+1}, \Delta'_{n+1}), \ldots (2)$$

   dans laquelle pour tout entier $l \in \{q + 1, \ldots, n + 1\}$:

– $S'_l = S_l \setminus \{s_1, \ldots, s_m\}$;
– $O'_l = O_l \setminus \{o_1, \ldots, o_{m'}\}$;

– $\theta'_l(x) = \begin{cases} s_0 & \text{si } \theta_l(x) \in \{s_1, \ldots, s_m\} \\ \theta_l(x) & \text{sinon} ; \end{cases}$

– $\theta'_l(y) = \begin{cases} o_0 & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'}\} \\ \theta_l(y) & \text{sinon} ; \end{cases}$

– $P'_l[s,o] = \begin{cases} P_l[s,o] \bigcup \cup_{t=1}^{m} P_l[s_t, o] & \text{si } s = s_0 \text{ et } o \neq o_0 \\ P_l[s,o] \bigcup \cup_{t=1}^{m'} P_l[s, o_t] & \text{si } o = o_0 \text{ et } s \neq s_0 \\ P_l[s,o] \bigcup \cup_{k=1}^{m} \cup_{t=1}^{m'} P_l[s_k, o_t] & \text{si } o = o_0 \text{ et } s = s_0 \\ P_l[s,o] & \text{sinon} ; \end{cases}$

où $s_0 \in S_0$ et $o_0 \in O_0$ sont fixés.

Le lecteur vérifiera que les assignations et les commandes de cette séquence permettent de passer de Δ_0 à Δ'_{n+1} car $P_l[s,o] \subseteq P'_l[s,o]$ pour tout entier $l \in \{q+1, \ldots, n+1\}$, pour tout sujet $s \in S'_l$ et pour tout objet $o \in O'_l$. Si le droit r apparaît dans Δ_{n+1} à v_{n+1} alors il apparaîtra dans Δ'_{n+1} à v_{n+1}. Nous avons donc pu construire une séquence de longueur inférieure à celle de la séquence minimale, ce qui est absurde. $\dashv$

Remarque : Nous notons par, "introduire r_1 dans $P[x_1, y_1]$, "introduire r_2 dans $P[x_2, y_2]$",...,"introduire r_{m_l} dans $P[x_{m_l}, y_{m_l}]$ ", les opérations primitives de la forme "introduire r' dans $P[x,y]$", de la commande α_l, $l \in \{1, \ldots, n\}$.

(b) Pour tout entier $l \in \{1, \ldots, n\}$, il existe un entier $t > l$ et un entier $f \in \{1, \ldots, m_l\}$ tel que la commande α_t a sa condition "$r_t \in P[x,y]$" qui vérifie $r_t = r_f$ et $(\theta_t(x), \theta_t(y)) = (\theta_l(x_f), \theta_l(y_f))$:

Preuve:
Soit une commande α_q, $q \in \{1, \ldots, n\}$ tel que pour tout entier $t > q$ et tout entier $f \in \{1, \ldots m_q\}$, la commande α_t a sa condition "$r_t \in P[x,y]$" qui vérifie $r_t \neq r_f$ ou $(\theta_t(x), \theta_t(y)) \neq (\theta_q(x_f), \theta_q(o_f))$. Comme précédemment considérons la séquence :

$$\ldots, (v_0, \Delta_0) \longrightarrow_{\alpha_1}^{\theta_1} (v_1, \Delta_1) \longrightarrow_{\alpha_2}^{\theta_2} \ldots \longrightarrow_{\alpha_{q-1}}^{\theta_{q-1}}$$
$$(v_{q-1}, \Delta_{q-1}) \longrightarrow_{\alpha_{q+1}}^{\theta'_{q+1}} (v_{q+1}, \Delta'_{q+1}) \longrightarrow_{\alpha_{q+2}}^{\theta'_{q+2}}$$

$$(v_{q+2}, \Delta'_{q+2}) \ldots \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} (v_{n+1}, \Delta'_{n+1}), \ldots (3)$$

dans laquelle pour tout entier $l \in \{q+1, \ldots, n+1\}$:

- $S'_l = S_l \setminus \{s_1, \ldots, s_m\}$;
- $O'_l = O_l \setminus \{o_1, \ldots, o_{m'}\}$;
- $\theta'_l(x) = \begin{cases} s_0 & \text{si } \theta_l(x) \in \{s_1, \ldots, s_m\} \\ \theta_l(x) & \text{sinon} ; \end{cases}$
- $\theta'_l(y) = \begin{cases} o_0 & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'}\} \\ \theta_l(y) & \text{sinon} ; \end{cases}$

où $s_0 \in S_0$, $o \in O_0$ sont fixés et où $s_1, \ldots, s_m$ et $o_1, \ldots, o_{m'}$ sont les sujets et les objets créés par la commande α_q. Nous avons ainsi pu construire une séquence de longueur inférieure à celle de la séquence de longueur minimale, ce qui est absurde. Par conséquent, toutes les commandes α_l de la séquence minimale contiennent des opérations primitives de la forme "introduire r' dans $P[x, y]$", $r' \neq r$, telles qu'il existe $t > l$ qui a sa condition de la forme "$r_t \in P[x, y]$", où $r_t = r'$ et $(\theta_t(x), \theta_t(y)) = (\theta_l(x), \theta_l(y))$. Nous savons aussi par hypothèse que toute commande $\alpha_l, 1 \leq l \leq n$ a sa condition "$r_l \in P[x, y]$" qui vérifie $\theta_l(x) \in S_0$ et $\theta_l(y) \in O_0$. Le nombre de commandes dans la séquence (1) qui permet d'arriver à l'apparition est donc au plus égal à $|S_0| \times |O_0| \times (|R| - 1)$+1. $\dashv$

2. $s_* \notin S_0$ ou $o_* \notin O_0$:

Notons qu'on a besoin d'au plus deux commandes pour créer une nouvelle cellule. Considérons le cas où $s_* \notin S_0$ et $o_* \notin O_0$. Le cas où $s_* \notin S_0$ et $o_* \in O_0$ et le cas où $s_* \in S_0$ et $o_* \notin O_0$ se traiteraient de la même façon. Soit $\alpha_b, 1 \leq b \leq n+1$, la commande qui contient l'opération primitive "créer le sujet x" avec $\theta_b(x) = s_*$ et $\alpha_c, 1 \leq c \leq n+1$, celle qui contient l'opération primitive "créer l'objet y" avec $\theta_c(y) = o_*$. Sans perte de généralité nous considèrons que $b < c$.

Il est facile de montrer, comme précédement, que toutes les commandes α_l $l \neq b$ et $l \neq c$ de la séquence minimale contiennent des opérations primitives de la forme "introduire r' dans $P[x, y]$", $r' \neq r$, telles qu'il existe $t > l$ qui a sa condition de la forme "$r_t \in P[x, y]$", où $r_t = r'$ et $(\theta_t(x), \theta_t(y)) = (\theta_l(x), \theta_l(y))$. Nous savons aussi par hypothèse que toute commande $\alpha_l, 1 \leq l \leq n$, a sa condition "$r_l \in P[x, y]$" qui vérifie

$\theta_l(x) \in S_0, \theta_l(y) \in O_0$. Le nombre de commandes différentes de α_b et α_c est au plus égal à $|S_0| \times |O_0| \times (|R| - 1)$. On peut donc borner le nombre $n + 1$ de commandes par $|S_0| \times |O_0| \times (|R| - 1)$+3.

⊣

Considérons maintenant le cas plus général dans lequel il existe dans la séquence qui fait apparaître r à un instant v donné des conditions qui portent sur de nouvelles cellules.

Lemme 6 *Soit* Π *un système de protection temporisée, spécifique, monotone et monoconditionnel,* r *un droit,* v *un nombre réel positif,* Δ *un état de protection et* h *une histoire,* $h \models \Pi$*, de séquence discontinue :*

$$\ldots (v_0, \Delta_0) \longrightarrow_{\alpha_1}^{\theta_1} (v_1, \Delta_1) \longrightarrow_{\alpha_2}^{\theta_2} \ldots \longrightarrow_{\alpha_n}^{\theta_n} (v_n, \Delta_n) \longrightarrow_{\alpha_{n+1}}^{\theta_{n+1}} (v_{n+1}, \Delta_{n+1}) \ldots (4)$$

dans laquel $\Delta_0 = \Delta$, $\Delta_{-1} = \Delta$, ... *et où* $v = v_{n+1}$. *Supposons que* h *fait apparaître* r *à* v_{n+1} *en utilisant un nombre minimum de commandes. Soit* $d \geq 0$ *le nombre de commandes dans* $\{\alpha_1, \ldots, \alpha_{n+1}\}$ *dont le test porte sur une cellule qui n'est pas initialement dans* $\Delta_0 = \Delta$. *On a :* $n \leq |S_0| \times |O_0| \times (|R - 1|) + 3d + 2$.

Preuve: Soit $\alpha_{k_1}, \alpha_{k_2}, ..., \alpha_{k_d}$, où $1 \leq k_1$ et $k_d \leq n + 1$, des commandes avec leur condition de la forme "$r_{k_l} \in P[x, y]$" telles que $\theta_{k_l}(x) \notin S_0$ ou $\theta_{k_l}(y) \notin O_0$. Pour tout entier $l \in \{1, \ldots, d\}$ on a :

- si $\theta_{k_l}(x) \notin S_0$ alors soit α_{m_l} la commande qui contient l'opération primitive "créer le sujet x" avec $\theta_{m_l}(x) = \theta_{k_l}(x)$;
- si $\theta_{k_l}(y) \notin O_0$ alors soit $\alpha_{m'_l}$ la commande qui contient l'opération primitive "créer l'objet y" avec $\theta_{m'_l}(y) = \theta_{k_l}(y)$;
- soit $\alpha_{m''_l}$ la commande qui contient l'opération primitive "introduire r_{k_l} dans $P[x, y]$" avec $\theta_{m''_l}(x) = \theta_{k_l}(x), \theta_{m''_l}(y) = \theta_{k_l}(y)$.

Notons que $m_l, m'_l, m''_l \in \{1, ..., k_l - 1\}$. Sans perte de généralité, nous supposons que $m_l \leq m'_l \leq m''_l$.

Remarque : Dans ce qui suit, $\theta_{k_1}(x)$ sera noté s_{k_1} et $\theta_{k_1}(y)$ sera noté o_{k_1}.

Soit (s_*, o_*), la cellule dans laquelle le droit r apparaît. On a besoin d'au plus deux commandes pour créer une nouvelle cellule. Sans perte de généralité, considérons dans cette démonstration le cas où $s_* \notin S_0$ et $o_* \notin O_0$. Soit $\alpha_b, 1 \leq b \leq n+1$, la commande qui contient l'opération primitive "créer le sujet x" avec $\theta_b(x) = s_*$

et $\alpha_c, 1 \leq c \leq n+1$, celle qui contient l'opération primitive "créer l'objet y" avec $\theta_c(y) = o_*$. Sans perte de généralité, nous considèrons que $b < c$.

Démontrons le fait suivant :

Fait : Pour tout entier $l \notin \{m_1, \ldots, m_d, m'_1, \ldots, m'_d, m''_1, \ldots, m''_d, b, c, n+1\}$, les commandes α_l ont des opérations primitives de la forme "introduire r' dans $P[x, y]$", $r' \neq r$, pour lesquelles il existe $t > l$ tel que la commande α_t a sa condition de la forme "$r_t \in P[x, y]$", où $r_t = r'$, $(\theta_t(x), \theta_t(y)) = (\theta_l(x), \theta_l(y))$, $\theta_l(x) \in S_0$ et $\theta_l(y) \in O_0$.

Si la séquence minimale contient une commande α_q qui ne satisfait pas cette propriété alors on considère la séquence :

$$\ldots, (v_0, \Delta_0) \underset{\alpha_1}{\overset{\theta_1}{\longrightarrow}} (v_1, \Delta_1) \underset{\alpha_2}{\overset{\theta_2}{\longrightarrow}} \ldots \underset{\alpha_{q-1}}{\overset{\theta_{q-1}}{\longrightarrow}} (v_{q-1}, \Delta_{q-1}) \underset{\alpha_{q+1}}{\overset{\theta'_{q+1}}{\longrightarrow}}$$
$$(v_{q+1}, \Delta'_{q+1}) \underset{\alpha_{q+2}}{\overset{\theta'_{q+2}}{\longrightarrow}} (v_{q+2}, \Delta'_{q+2}) \ldots \underset{\alpha_{n+1}}{\overset{\theta'_{n+1}}{\longrightarrow}} (v_{n+1}, \Delta'_{n+1}), \ldots (5)$$

dans laquelle pour tout entier $l \in \{q+1, \ldots, n+1\}$ on a :

- $S'_l = S_l \setminus \{s_1, \ldots, s_m\}$;
- $O'_l = O_l \setminus \{o_1, \ldots, o_{m'}\}$;
- $\theta'_l(x) = \begin{cases} s_0 & \text{si } \theta_l(x) \in \{s_1, \ldots, s_m\} \\ \theta_l(x) & \text{sinon ;} \end{cases}$
- $\theta'_l(y) = \begin{cases} o_0 & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'}\} \\ \theta_l(y) & \text{sinon.} \end{cases}$

où $s_0 \in S_0$ et $o_0 \in O_0$ sont fixés et où $s_1, \ldots, s_m$ et $o_1, \ldots, o_{m'}$ sont les sujets et les objets créés par la commande α_q. Notons que, dans cette séquence (5), le droit r apparaît bien à v_{n+1}. Nous avons donc pu trouver une séquence qui fait apparaître r à v_{n+1} de longueur inférieur à celle de la séquence minimale. On a donc une contradiction.

La séquence minimale contient ainsi au plus $|S_0| \times |O_0| \times (|R| - 1) + 3d + 3$ commandes. Ceci termine la preuve du lemme 6. $\dashv$

Il suffira donc de borner d pour borner la séquence minimale. Pour ce faire, nous allons utiliser les résultats des lemmes 7 et 8 suivants.

Lemme 7 *Soit Π un système de protection temporisée, spécifique, monotone et monoconditionnel, r un droit, v un nombre réel positif, Δ un état de protection et h une histoire, $h \models \Pi$, de séquence discontinue :*

$$\ldots (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_n}_{\alpha_n} (v_n, \Delta_n) \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} (v_{n+1}, \Delta_{n+1}) \ldots (4)$$

où $\Delta_0 = \Delta$, $\Delta_{-1} = \Delta$, ... et où $v = v_{n+1}$. Supposons que h fait apparaître r à v_{n+1} en utilisant un nombre minimum de commandes. Soit α_{k_1},$k_1 \in \{1, \ldots, n+1\}$ la première commande qui a sa condition de la forme "$r_{k_1} \in P[x, y]$" et telle que $\theta_{k_1}(x) \notin S_0$ ou $\theta_{k_1}(y) \notin O_0$. Soit $\alpha_{m''_1}$ la commande qui contient l'opération primitive "introduire r_{k_1} dans $P[x, y]$" telle que $\theta_{m''_1}(x) = \theta_{k_1}(x)$ et $\theta_{m''_1}(y) = \theta_{k_1}(y)$. On a :

- *une chaîne de $r_{m''_1+2} \in (s_{m''_1+2}, o_{m''_1+2})$ à $r_{n+1} \in (s_{n+1}, o_{n+1})$.*

Preuve:

Raisonnant par l'absurde, soit $t \in \{m''_1 + 2, \ldots, n + 1\}$ le plus grand entier tel que, la commande α_t a sa condition "$r_t \in P[x, y]$" et est telle qu'il existe un entier $q < t - 1$ tel que $r_t \in P_q[\theta_t(x), \theta_t(x)]$. Considérons le plus petit entier q qui satisfait ce critère. Il existe deux cas possibles pour q : $q < m''_1$ et $q \geq m''_1$.

1. $q < m''_1$: Soit la séquence minimale

$$\ldots, (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} \ldots \longrightarrow^{\theta_q}_{\alpha_q} (v_q, \Delta_q) \longrightarrow^{\theta_{q+1}}_{\alpha_{q+1}} \ldots \longrightarrow^{\theta_{m''_1}}_{\alpha_{m''_1}}$$
$$(v_{m''_1}, \Delta_{m''_1}) \longrightarrow^{\theta_{m''_1+1}}_{\alpha_{m''_1+1}} \ldots \longrightarrow^{\theta_{t-1}}_{\alpha_{t-1}} (v_{t-1}, \Delta_{t-1}) \longrightarrow^{\theta_t}_{\alpha_t} (v_t, \Delta_t) \longrightarrow^{\theta_{t+1}}_{\alpha_{t+1}}$$
$$\ldots \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} (v_{n+1}, \Delta_{n+1}), \ldots (6)$$

   La commande α_{n+1} qui fait apparaître r dans Δ_{n+1} contient nécessairement l'opération primitive "introduire r dans $P[x, y]$" telle que $\theta(x) = s_*$ et $\theta(y) = o_*$. On distingue quatre cas :

   - Cas 1 : La commande α_{n+1} contient l'opération primitive "créer le sujet x" et ne contient pas l'opération primitive "créer l'objet y".
   - Cas2 : La commande α_{n+1} contient l'opération primitive "créer le sujet x" et l'opération primitive "créer l'objet y".
   - Cas 3 : La commande α_{n+1} ne contient pas l'opération primitive "créer le sujet x" et contient l'opération primitive "créer l'objet y".
   - Cas 4 : La commande α_{n+1} ne contient ni l'opération primitive "créer le sujet x" ni l'opération primitive "créer l'objet y".

Considérons la séquence :

$$\ldots,(v_0,\Delta_0)\longrightarrow^{\theta_1}_{\alpha_1}\ldots\longrightarrow^{\theta_q}_{\alpha_q}(v_q,\Delta_q)\longrightarrow^{\theta_{q+1}}_{\alpha_{q+1}}\ldots\longrightarrow^{\theta''_{m''_1}}_{\alpha_{m''_1}}$$
$$(v_{m''_1},\Delta_{m''_1})\longrightarrow^{\theta'_t}_{\alpha'_t}(v_t,\Delta'_t)\longrightarrow^{\theta'_{t+1}}_{\alpha_{t+1}}\ldots\longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}}(v_{n+1},\Delta'_{n+1}),\ldots(7)$$

telle que pour tout entier $l\in\{t,\ldots,n+1\}$:

- $S'_l = S_l\setminus\{s_1,\ldots,s_m\}$;
- $O'_l = O_l\setminus\{o_1,\ldots,o_{m'}\}$;

et telle que :

(a) si nous sommes dans le cas 1 alors :

- $\theta'_l(x)=\begin{cases} s_{k_1} & \text{si } \theta_l(x)\in\{s_1,\ldots,s_m\}\\ \theta_l(x) & \text{sinon ;}\end{cases}$
- $\theta'_l(y)=\begin{cases} o_{k_1} & \text{si } \theta_l(y)\in\{o_1,\ldots,o_{m'+1}\}\\ \theta_l(y) & \text{sinon ;}\end{cases}$

(b) si nous sommes dans le cas 2 alors :

- $\theta'_l(x)=\begin{cases} s_{k_1} & \text{si } \theta_l(x)\in\{s_1,\ldots,s_m\}\\ \theta_l(x) & \text{sinon ;}\end{cases}$
- $\theta'_l(y)=\begin{cases} o_{k_1} & \text{si } \theta_l(y)\in\{o_1,\ldots,o_{m'}\}\\ \theta_l(y) & \text{sinon ;}\end{cases}$

(c) si nous sommes dans le cas 3 alors :

- $\theta'_l(x)=\begin{cases} s_{k_1} & \text{si } \theta_l(x)\in\{s_1,\ldots,s_{m+1}\}\\ \theta_l(x) & \text{sinon ;}\end{cases}$
- $\theta'_l(y)=\begin{cases} o_{k_1} & \text{si } \theta_l(y)\in\{o_1,\ldots,o_{m'}\}\\ \theta_l(y) & \text{sinon ;}\end{cases}$

(d) si nous sommes dans le cas 4 alors :

- $\theta'_l(x)=\begin{cases} s_{k_1} & \text{si } \theta_l(x)\in\{s_1,\ldots,s_{m+1}\}\\ \theta_l(x) & \text{sinon ;}\end{cases}$
- $\theta'_l(y)=\begin{cases} o_{k_1} & \text{si } \theta_l(y)\in\{o_1,\ldots,o_{m'+1}\}\\ \theta_l(y) & \text{sinon ;}\end{cases}$

où $s_1, \ldots, s_m$ et $o_1, \ldots, o_{m'}$ sont les sujets et les objets créés par les commandes supprimées α_h, $m_1'' + 1 \leq h \leq t-1$, et où $(s_{m+1}, o_{m'+1})=(s_*, o_*)$ est la cellule dans laquelle le droit r apparaît dans la séquence minimale. Les assignations et les commandes permettent dans les quatre cas de passer de Δ_0 à Δ'_{n+1}. De même, dans les quatre cas possibles pour la commande α_{n+1}, si le droit r apparaît dans Δ_{n+1} à v_{n+1} alors il apparaîtra dans Δ'_{n+1} à v_{n+1}. Plus précisement, dans la cellule (s_*, o_{k_1}) dans le cas 1, dans la cellule (s_*, o_*) dans le cas 2, dans la cellule (s_{k_1}, o_*) dans la cas 3 et dans la cellule (s_{k_1}, o_{k_1}) dans le cas 4. Nous avons pu construire une séquence qui a un nombre de commandes inférieur à celui de la séquence minimale, ce qui est absurde.

2. $q \geq m_1''$: Soit la séquence minimale

$$\ldots, (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} \ldots \longrightarrow^{\theta_{m_1''}}_{\alpha_{m_1''}} (v_{m_1''}, \Delta_{m_1''}) \longrightarrow^{\theta_{m_1''+1}}_{\alpha_{m_1''+1}} \ldots \longrightarrow^{\theta_q}_{\alpha_q}$$
$$(v_q, \Delta_q) \longrightarrow^{\theta_{q+1}}_{\alpha_{q+1}} \ldots \longrightarrow^{\theta_{t-1}}_{\alpha_{t-1}} (v_{t-1}, \Delta_{t-1}) \longrightarrow^{\theta_t}_{\alpha_t} (v_t, \Delta_t) \longrightarrow^{\theta_{t+1}}_{\alpha_{t+1}}$$
$$\ldots \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} (v_{n+1}, \Delta_{n+1}), \ldots (8)$$

Nous supposons comme dans (a), qu'il existe quatre cas possibles pour la commande α_{n+1}. Considérons la séquence :

$$\ldots, (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} \ldots \longrightarrow^{\theta_{m_1''}}_{\alpha_{m_1''}} (v_{m_1''}, \Delta_{m_1''}) \longrightarrow^{\theta_{m_1''+1}}_{\alpha_{m_1''+1}} \ldots \longrightarrow^{\theta_q}_{\alpha_q}$$
$$(v_q, \Delta_q) \longrightarrow^{\theta'_t}_{\alpha_t} (v_t, \Delta'_t) \longrightarrow^{\theta'_{t+1}}_{\alpha_{t+1}} \ldots \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} (v_{n+1}, \Delta'_{n+1}), \ldots (9)$$

telle que pour tout entier $l \in \{t, \ldots, n+1\}$, la construction des S'_l, O'_l, $\theta_l(x)$ et $\theta_l(y)$ se fait exactement comme dans (a) sauf que $s_1, \ldots, s_m$ et $o_1, \ldots, o_{m'}$ vont repésenter les sujets et les objets créés par les commandes supprimées par α_h, $q+1 \leq h \leq t-1$. Les assignations et les commandes permettent de passer de Δ_0 à Δ'_{n+1}. Comme dans (a), dans les quatre cas possibles pour la commande α_{n+1}, si le droit r apparaît dans Δ_{n+1} à v_{n+1} alors il apparaîtra dans Δ'_{n+1} à v_{n+1}. Plus précisement, dans la cellule (s_*, o_{k_1}) dans le cas 1, dans la cellule (s_*, o_*) dans le cas 2, dans la cellule (s_{k_1}, o_*) dans la cas 3 et dans la cellule (s_{k_1}, o_{k_1}) dans le cas 4. Nous avons ainsi pu construire une séquence qui a un nombre de commandes inférieur à celui de la séquence minimale, ce qui est absurde.

On a donc une chaîne de $r_{m_1''+2} \in (s_{m_1''+2}, o_{m1_1''+2})$ à $r_{n+1} \in (s_{n+1}, o_{n+1})$.

⊣

Nous allons utiliser le résultat du lemme 7 pour démontrer le lemme 8 ci-dessous. Ce qui nous permettra de borner d et par conséquent de borner la longueur de la séquence minimale qui fait apparaître le droit r à un instant v donné.

Lemme 8 *Soit Π un système de protection temporisée, spécifique, monotone et monoconditionnel, r un droit, v un nombre réel positif, Δ un état de protection et h une histoire, $h \models \Pi$, de séquence discontinue :*

$$\dots (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \dots \longrightarrow^{\theta_n}_{\alpha_n} (v_n, \Delta_n) \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} (v_{n+1}, \Delta_{n+1}) \dots (4)$$

où $\Delta_0 = \Delta$, $\Delta_{-1} = \Delta$, ... et où $v = v_{n+1}$. Supposons que h fait apparaître r à v_{n+1} en utilisant un nombre minimum de commandes. Soit $\alpha_{k_1}, \alpha_{k_2}, ..., \alpha_{k_d}$, où $1 \leq k_1$ et $k_d \leq n+1$, des commandes avec leur condition de la forme "$r_{k_l} \in P[x, y]$" telles que $\theta_{k_l}(x) \notin S_0$ ou $\theta_{k_l}(y) \notin O_0$. On a :

– *$\forall l \neq q$, $r_{k_l} \neq r_{k_q}$ où $l, q \in \{1, \dots, d\}$.*

Preuve:

Soit α_{k_1},$k_1 \in \{1, \dots, n+1\}$, la première commande qui a sa condition de la forme "$r_{k_1} \in p[x, y]$" telle que $\theta_{k_1}(x) \notin S_0$ ou $\theta_{k_1}(y) \notin O_0$. Soit $\alpha_{m''_1}$ la commande qui contient l'opération primitive "introduire r_{k_1} dans $P[x, y]$" telle que $\theta_{m''_1}(x) = \theta_{k_1}(x)$ et $\theta_{m''_1}(y) = \theta_{k_1}(y)$. Supposons qu'il existe deux commandes α_{k_t} et α_{k_q} dont les conditions sont respectivement "$r_{k_t} \in p[x, y]$" et "$r_{k_q} \in p[x, y]$" avec :

– $\theta_{k_t}(x) \notin S_0$ ou $\theta_{k_t}(y) \notin O_0$ et
– $\theta_{k_q}(x) \notin S_0$ ou $\theta_{k_q}(y) \notin O_0$.

Supposons aussi que $r_{k_t} = r_{k_q}$

Remarque : Nous noterons respectivement $\theta_{k_t}(x), \theta_{k_q}(x), \theta_{k_t}(y)$ et $\theta_{k_q}(y)$ par $s_{k_t}, s_{k_q}, o_{k_t}$ et o_{k_q}.

Soit la séquence minimale :

$$\dots, (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} \dots \longrightarrow^{\theta_{m''_1}}_{\alpha_{m''_1}} (v_{m''_1}, \Delta_{m''_1}) \longrightarrow^{\theta_{m''_1+1}}_{\alpha_{m''_1+1}} \dots \longrightarrow^{\theta_{k_t}}_{\alpha_{k_t}}$$
$$(v_{k_t}, \Delta_{k_t}) \longrightarrow^{\theta_{k_t+1}}_{\alpha_{k_t+1}} \dots \longrightarrow^{\theta_{k_q}}_{\alpha_{k_q}} (v_{k_q}, \Delta_{k_q}) \longrightarrow^{\theta_{k_q+1}}_{\alpha_{k_q+1}} \dots \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}}$$
$$(v_{n+1}, \Delta_{n+1}), \dots (10)$$

Nous supposons comme précédemment, qu'il existe quatre cas pour la commande α_{n+1}. Considérons la séquence :

$$\ldots,(v_0,\Delta_0) \overset{\theta_1}{\underset{\alpha_1}{\longrightarrow}} \ldots \overset{\theta_{m_1''}}{\underset{\alpha_{m_1''}}{\longrightarrow}} (v_{m_1''},\Delta_{m_1''}) \overset{\theta_{m_1''+1}}{\underset{\alpha_{m_1''+1}}{\longrightarrow}} \ldots \overset{\theta_{k_t-1}}{\underset{\alpha_{k_t-1}}{\longrightarrow}}$$
$$(v_{k_t-1},\Delta_{k_t-1}) \overset{\theta'_{k_q}}{\underset{\alpha_{k_q}}{\longrightarrow}} (v_{k_q},\Delta'_{k_q}) \overset{\theta'_{k_q+1}}{\underset{\alpha_{k_q+1}}{\longrightarrow}} \ldots \overset{\theta'_{n+1}}{\underset{\alpha_{n+1}}{\longrightarrow}} (v_{n+1},\Delta'_{n+1}),\ldots (11)$$

dans laquelle pour tout entier $l \in \{k_q,\ldots,n+1\}$:

- $S'_l = S_l \setminus \{s_1,\ldots,s_m\}$;
- $O'_l = O_l \setminus \{o_1,\ldots,o_{m'}\}$;

et dans laquelle :

1. si nous sommes dans le cas 1 alors :
   - $\theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1,\ldots,s_m\} \setminus \{s_{k_q}\} \\ s_{k_t} & \text{si } \theta_l(x) = s_{k_q} \\ \theta_l(x) & \text{sinon ;} \end{cases}$
   - $\theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1,\ldots,o_{m'+1}\} \setminus \{o_{k_q}\} \\ o_{k_l} & \text{si } \theta_l(y) = o_{k_q} \\ \theta_l(y) & \text{sinon ;} \end{cases}$
2. si nous sommes dans le cas 2 alors :
   - $\theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1,\ldots,s_m\} \setminus \{s_{k_q}\} \\ s_{k_t} & \text{si } \theta_l(x) = s_{k_q} \\ \theta_l(x) & \text{sinon ;} \end{cases}$
   - $\theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1,\ldots,o_{m'}\} \setminus \{o_{k_q}\} \\ o_{k_l} & \text{si } \theta_l(y) = o_{k_q} \\ \theta_l(y) & \text{sinon ;} \end{cases}$
3. si nous sommes dans le cas 3 alors :
   - $\theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1,\ldots,s_{m+1}\} \setminus \{s_{k_q}\} \\ s_{k_t} & \text{si } \theta_l(x) = s_{k_q} \\ \theta_l(x) & \text{sinon ;} \end{cases}$
   - $\theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1,\ldots,o_{m'}\} \setminus \{o_{k_q}\} \\ o_{k_l} & \text{si } \theta_l(y) = o_{k_q} \\ \theta_l(y) & \text{sinon ;} \end{cases}$
4. si nous sommes dans le cas 4 alors :

$$- \ \theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1, \ldots, s_{m+1}\} \setminus \{s_{k_q}\} \\ s_{k_t} & \text{si } \theta_l(x) = s_{k_q} \\ \theta_l(x) & \text{sinon ;} \end{cases}$$

$$- \ \theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'+1}\} \setminus \{o_{k_q}\} \\ o_{k_l} & \text{si } \theta_l(y) = o_{k_q} \\ \theta_l(y) & \text{sinon ;} \end{cases}$$

où $s_1, \ldots, s_m$ et $o_1, \ldots, o_{m'}$ sont les sujets et les objets créés par les commandes supprimées $\alpha_h, k_t \leq h \leq k_{q-1}$ et où $(s_{m+1}, o_{m'+1}) = (s_*, o_*)$ est la cellule dans laquelle le droit r apparaît dans la séquence minimale. Les assignations et les commandes permettent de passer de Δ_0 à Δ'_{n+1}, d'une part grâce au fait d'avoir une chaîne de $r_{m''_1+2} \in (s_{m''_1+2}, o_{m''_1+2})$ à $r_{n+1} \in (s_{n+1}, o_{n+1})$ et d'autre part grâce au fait d'appliquer les commandes α_l, $l \in \{k_q, \ldots, n+1\}$, à v_l dans la séquence (11). On peut facilement vérifier, dans les quatre cas possibles pour la commande α_{n+1}, que si r apparaît dans Δ_{n+1} à v_{n+1} alors il apparaîtra dans Δ'_{n+1} à v_{n+1}. Plus précisement, dans la cellule (s_*, o_{k_1}) dans le cas 1, dans la cellule (s_*, o_*) dans le cas 2, dans la cellule (s_{k_1}, o_*) dans la cas 3 et dans la cellule (s_{k_1}, o_{k_1}) dans le cas 4. Nous avons pu construire une séquence ayant un nombre de commandes inférieur à celui de la séquence (4), ce qui est absurde. On ne peut donc pas avoir deux commandes qui testent le même droit dans des nouvelles cellules. Ceci termine la preuve du lemme 8. $\dashv$
Par conséquent, avec les lemmes 6 et 8, on en conclut que pour un instant v donné, si

$$\ldots (v_0, \Delta_0) \longrightarrow^{\theta_1}_{\alpha_1} (v_1, \Delta_1) \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_n}_{\alpha_n} (v_n, \Delta_n) \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} (v_{n+1}, \Delta_{n+1}) \ldots$$

est une séquence discontinue qui fait apparaître r à $v_{n+1} = v$, en utilisant un nombre minimum de commandes, alors $n \leq (|S_0| \times |O_0| + 3) \times (|R| - 1) + 2$.

Le fait de borner le nombre de commandes nous permet de construire un algorithme qui testera toute les combinaisons possibles de commandes. Cela nous permet également de trouver la séquence discontinue qui fera apparaître r le plus tôt. On pourra ainsi déterminer la valeur du plus petit nombre réel positif d pour lequel Π est d-attaqué. Ceci termine la preuve du théoreme 9. $\dashv$

Nous avons donc pu montrer que le problème de la PROTECTION TEMPORISEE MINIMALE($\mathcal{C}^+(1, +\infty)$) est décidable.

Conclusion

La sécurité informatique traite de la prévention et de la détection d'actions non autorisées par les utilisateurs d'un système informatique. Le contrôle d'accès est l'un de ses concept fondamentaux. Nous avons commencé dans ce rapport par donner un aperçu sur les différents travaux réalisés dans le domaine du contrôle d'accès. Ensuite nous avons introduit les systèmes de protection temporisée où des nombres réels sont utilisés pour représenter le temps et où l'état de protection est défini par trois matrices : matrice d'actions, matrice de buts et matrice de permissions. Grâce aux systèmes de protection temporisée nous avons pu définir le concept de la disponibilité, trois problèmes différents de la disponibilité, ainsi que trois autres problèmes de la protection temporisée. Le manque de temps ne nous a pas permis de donner des résultats concernant la décidabilité de ces six problèmes. Nous avons quand même pu établir des résultats mathématiques concernant la décidabilité des trois problèmes de la protection temporisée dans le cadre restreint des systèmes de protection temporisée spécifiques où l'état de protection est représenté par une seule matrice, celles des permissions et où les conditions temporisées sont de la forme "$r \in P[s, o]$ depuis au moins la durée d".

Dans le cadre des systèmes de protection temporisée spécifiques, il faudrait compléter les résultats obtenus dans ce rapport en étudiant le problème de la protection temporisée bornée, dans la classe la plus générale des systèmes de protection temporisée $\mathcal{C}(+\infty, +\infty)$ et pour la classe des systèmes de protection temporisée monotones $\mathcal{C}^+(+\infty, +\infty)$. Ensuite on pourrait considérer d'autres systèmes de protection temporisée spécifiques : ceux dans lesquels les conditions temporisées des commandes seraient de la forme "$r \in P[s, o]$ depuis au plus la durée d" et ceux dans lesquels les conditions temporisées seraient de la forme "$r \notin P[s, o]$ depuis au moins la durée d". Il serait également intéressant d'introduire la notion de rôle dans les systèmes de protection temporisée pour réduire la taille de la matrice des permissions. Le but de tous ces travaux est donc d'arriver, à partir des différents systèmes de protection temporisée spécifiques, à utiliser les résultats obtenus concernant la décidabilité du problème de la protection temporisée pour s'at-

taquer aux systèmes de protection temporisée et ainsi résoudre les six problèmes posés dans ce cadre. Cependant un autre type de systèmes de protection temporisée peut être intéressant à considérer : celui qui contient des opérations primitives temporisées. C'est-à-dire que, dans ce type de systèmes de protection temporisée, les droits seraient accordés aux utilisateurs sur les fichiers pour une durée fixée. Une fois cette durée écoulée, l'utilisateur viendrait à perdre le droit en question. L'évolution de la matrice d'accès, dans ce cas, se ferait non seulement grâce à des commandes mais aussi en fonction du temps.

Bibliographie

[1] A. A. Abou El Kalam, S. Benferhat, A. Miège, R. El Baida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, and P Trouessin. Organization based access control. In *POLICY*, pages 120–. IEEE Computer Society, 2003.

[2] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2) :123–154, 1984.

[3] P. Balbiani and F. Cheikh. Une approche uniforme de la modélisation des systèmes de protection temporisés. Journées Formalisation des Activités Concurrentes, march 2005.

[4] P. Balbiani, F. Harb, and A. Kaafarani. Access control with prohibitions and obligations. To appear, 2005.

[5] D. Bell and L. LaPadula. Secure computer systems. unified exposition and multics interpretation. Technical Report MTR-2297, The MITRE Corporation, 1975.

[6] K. Biba. Integrity considerations for secure computer systems. Technical Report MTR-3153, The MITRE Corporation, 1977.

[7] M. Bishop. *Computer Security*. Addison Wesley, 2003.

[8] J. Broersen, F. Dignum, V. Dignum, and JJ. C. Meyer. Designing a deontic logic of deadlines. In A. Lomuscio and D. Nute, editors, *DEON*, volume 3065 of *Lecture Notes in Computer Science*, pages 43–56. Springer, 2004.

[9] T. Budd and R. Lipton. Inert rights and conspirators in the take-grant system. Technical Report 126, Department of Computer Science, Yale University, 1977.

[10] T. Budd and R. Lipton. On classes of protection systems. In *Proceedings of the Workshop on Foundations of Computer Security*, pages 281–296. Academic Press, 1978.

[11] J. Happen. Reasoning about knowledge : An overview. In *Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 1–17. Morgan Kaufmann, 1986.

[12] M. Harrison and W. Ruzzo. Monotonic protection systems. In *Proceedings of the Workshop on Foundations of Computer Security*, pages 337–363. Academic Press, 1978.

[13] M. A. Harrison. Theoretical issues concerning protection in operating systems. *Advances in Computers*, 24 :61–100, 1985.

[14] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8) :461–471, 1976.

[15] A. Herzig and O. Rifi. Update operations : A review. In *Proceedings of ECAI*, pages 13–17, 1998.

[16] A. K. Jones, R. J. Lipton, and L. Snyder. A linear time algorithm for deciding security. In *Proceedings of the 17th Conference on Foundations of Computer Science (FOCS'17)*, pages 33–41, 1976.

[17] B. Lampson. Protection. *Operating Systems Review*, 8 :18–24, 1974.

[18] R. J. Lipton and L. Snyder. On synchronisation and security. In *Proceedings of the Workshop on Foundations of Computer Security*, pages 367–385. Academic Press, 1978.

[19] P. Régnier. Algorithmique de la planification en ia. 2004.

[20] R. Sandhu. The typed access matrix model. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 122–136. IEEE Society, 1992.

[21] R. S. Sandhu. Transformation of access rights. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 259–268, 1989.

[22] R. S. Sandhu and S. Ganta. On testing for absence of rights in access control models. In *Proceedings of the Computer Security Foundations Workshop (CSFW)*, pages 109–118. IEEE Society, 1993.

[23] L. Snyder. Formal models of capability-based protection systems. *IEEE Transactions on Computers*, 30(3) :172–181, 1981.

Annexe A

décidabilité du problème de la protection pour la classe des systèmes de protection mono-opérationnels

Nous montrerons dans cette annexe que le problème de la protection est décidable, pour la classe des systèmes de protection mono-opérationnels. Pour cela nous allons proposer quatre lemmes avec leur démonstration.

Remarque :

1. Sans perte de généralité, nous supposons que lorsqu'un sujet ou un objet est créé dans une séquence $\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{m-1}}_{\alpha_{m-1}} \Delta_{m-1} \longrightarrow^{\theta_m}_{\alpha_m} \Delta_m$, c'est qu'il n'apparaît pas avant dans la séquence,
2. Dans un système de protection mono-opérationnel la forme d'une commande est égale à la forme de l'unique opération primitive qu'elle contient.

Lemme 9 *Soit Π un système de protection mono-opérationnel, r un droit, et Δ un état de protecion. Soit*

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{m-1}}_{\alpha_{m-1}}, \Delta_{m-1} \longrightarrow^{\theta_m}_{\alpha_m} \Delta_m \text{ (1)}$$

une séquence de longueur minimale telle que $\Delta_0 = \Delta$ et au terme de laquelle α_m fait apparaître r dans Δ_{m-1}. Pour tout entier $l \in \{1, \ldots, m-1\}$, aucune commande α_l n'est de la forme "détruire le sujet x", ou "détruire l'objet y".

Preuve:

Soit α_n la dernière commande de la forme "détruire le sujet x" avec $\theta_n(x) = s_n$ (respectivement "détruire l'objet y" avec $\theta_n(y) = o_n$).
Considérons la séquence :

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{n-1}}_{\alpha_{n-1}} \Delta_{n-1} \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} \Delta'_{n+1}$$

$$\longrightarrow^{\theta'_{n+2}}_{\alpha_{n+2}} \ldots \longrightarrow^{\theta'_m}_{\alpha_m} \Delta'_m \text{ (2)}$$

dans laquelle pour tout entier $l \in \{n+1, \ldots, m\}$:
- $S'_l = S_l \cup \{s_n\}$ (resp $S'_l = S_l$),
- $O'_l = O_l \cup \{s_n\}$ (resp $O'_l = O_l \cup \{o_n\}$),
- $\theta'_l(x) = \theta_l(x)$ pour toute variable x de type sujet,
- $\theta'_l(y) = \theta_l(y)$ pour toute variable y de type objet et
- $P'_l[s,o] = \begin{cases} P_{n-1}[s,o] & \text{si } s = s_n, s \in S_l \text{ ou } o = s_n, o \in O_l \\ & \text{(resp } o = o_n, o \in O_l\text{)} \\ P_l[s,o] & \text{sinon.} \end{cases}$

Le lecteur vérifiera que les assignations et les commandes de cette séquence, permettent de passer de Δ_0 à Δ'_m. La raison en est que les commandes ne testent pas l'absence d'un droit dans une cellule (on n'a pas de commande avec conditions négatives) et que $P_l[s,o] \subseteq P'_l[s,o]$, pour tout entier $l \in \{0, \ldots, n-1, n+1, \ldots, m\}$, pour tout sujet $s \in S_l$ et pour tout objet $o \in O_l$. Par conséquent, si la commande α_m fait apparaître r dans Δ_{m-1} alors elle fera également apparaître r dans Δ'_{m-1}. Nous avons ainsi pu construire une séquence de longueur $m-1$ qui est plus courte que la séquence de longueur minimale. Ce qui est absurde. On ne peut pas, de ce fait, utiliser une commande de la forme "détruire le sujet x" ou "détruire l'objet y" pour la construction d'une séquence minimale d'états de protection. $\dashv$

Lemme 10 *Soit Π un système de protection mono-opérationnel, r un droit, Δ un état de protection et*

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{m-1}}_{\alpha_{m-1}} \Delta_{m-1} \longrightarrow^{\theta_m}_{\alpha_m} \Delta_m \text{ (1)}$$

une séquence de longueur minimale telle que $\Delta_0 = \Delta$ et au terme de laquelle α_m fait apparaître r dans Δ_{m-1}. Il existe au plus une commande $\alpha_l, l \in \{1, \ldots, m\}$ de la forme "supprimer r_l de $P[x,y]$".

Preuve:

1. Il n'existe aucune commande de la forme "supprimer r' de $P[x,y]$" ($r' \neq r$).

   Preuve: Soit α_n la dernière commande de la forme "supprimer r_n de $P[x,y]$" ($r_n \neq r$) avec $\theta_n(x) = s_n$ et $\theta_n(y) = o_n$.
   Considérons la séquence suivante :

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{n-1}}_{\alpha_{n-1}} \Delta_{n-1} \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} \Delta'_{n+1}$$
$$\longrightarrow^{\theta'_{n+2}}_{\alpha_{n+2}} \ldots \longrightarrow^{\theta'_m}_{\alpha_m} \Delta'_m \text{ (3)}$$

   dans laquelle pour tout entier $l \in \{n+1, \ldots, m\}$:
   - $S'_l = S_l$,
   - $O'_l = O_l$,
   - $\theta'_l(x) = \theta_l(x)$ pour toute variable x de type sujet,
   - $\theta'_l(y) = \theta_l(y)$ pour toute variable y de type objet et

– $P'_l[s,o] = \begin{cases} P_l[s,o] \cup \{r'\} & \text{si } s = s_n \text{ et } o = o_n \\ P_l[s,o] & \text{sinon.} \end{cases}$

Il est facile de vérifier que les commandes et les assignations de cette séquence, permettent de passer de Δ_0 à Δ'_m. En effet, $P_l[s,o] \subseteq P'_l[s,o]$ pour tout entier $l \in \{0,\ldots,n-1,n+1,\ldots,m\}$, pour tout sujet $s \in S_l$ et pour tout objet $o \in O_l$. Si α_m fait apparaître r dans Δ_{m-1} alors elle le fera dans Δ'_{m-1}. Nous avons donc pu construire une séquence de longueur $m-1$ qui est plus courte que la séquence de longueur minimale, ce qui est absurde. Par conséquent on ne peut pas avoir dans la séquence de longueur minimale une commande de la forme "supprimer r' de $P[x,y]$" ($r' \neq r$).
⊣

2. Il existe au plus une commande de la forme "supprimer r de $P[x,y]$".

Preuve: Soit α_{n-q} et α_n deux commandes de la forme "supprimer r de $P[x,y]$" telles que :
– $\theta_{n-q}(x) = s_{n-q}$,
– $\theta_{n-q}(y) = o_{n-q}$,
– $\theta_n(x) = s_n$ et
– $\theta_n(y) = o_n$.

Remarque
Les cellules (s_{n-q}, o_{n-q}) et (s_n, o_n) sont distinctes. En effet,leur égalité contredirait la minimalité de la séquence.

Il existe deux possibilités pour l'apparition du droit r. Il peut apparaître dans une des deux cellules (s_{n-q}, o_{n-q}), (s_n, o_n) ou dans une cellule (s,o) différente des deux cellules précédentes.

(a) Le droit r apparaît dans la cellule (s_n, o_n) ou dans la cellule (s_{n-q}, o_{n-q}) :

Nous supposons que r apparaît dans la cellule (s_n, o_n) (nous avons le même argument si r apparaît dans (s_{n-q}, o_{n-q})). Considérons la séquence suivante :

$$\Delta_0 \underset{\alpha_1}{\overset{\theta_1}{\longrightarrow}} \Delta_1 \underset{\alpha_2}{\overset{\theta_2}{\longrightarrow}} \cdots \underset{\alpha_{n-q-1}}{\overset{\theta_{n-q-1}}{\longrightarrow}} \Delta_{n-q-1} \underset{\alpha_{n-q+1}}{\overset{\theta'_{n-q+1}}{\longrightarrow}} \Delta'_{n-q+1}$$
$$\underset{\alpha_{n-q+1}}{\overset{\theta'_{n-q+2}}{\longrightarrow}} \cdots \underset{\alpha_m}{\overset{\theta'_m}{\longrightarrow}} \Delta'_m \quad (4)$$

dans laquelle pour tout entier $l \in \{n-q+1,\ldots,m\}$:
– $S'_l = S_l$,
– $O'_l = O_l$,
– $\theta'_l(x) = \theta_l(x)$ pour toute variable x de type sujet,
– $\theta'_l(y) = \theta_l(y)$ pour toute variable y de type objet et

– $P'_l[s,o] = \begin{cases} P_l[s,o] \cup \{r\} & \text{si } s = s_{n-q} \text{ et } o = o_{n-q} \\ P_l[s,o] & \text{sinon.} \end{cases}$

Cette séquence a une commande de moins (α_{n-q}) que la séquence (1). Il est facile de vérifier que les commandes et les assignations de cette séquence, permettent de passer de Δ_0 à Δ'_m. En effet, $P_l[s,o] \subseteq P'_l[s,o]$ pour tout entier $l \in \{0,\ldots,n-q-1,n-q+1,\ldots,m\}$, pour tout sujet $s \in S_l$ et pour tout objet $o \in O_l$. Si la commande α_m fait apparaître le droit r dans Δ_{m-1} alors elle fera également apparaître r dans Δ'_{m-1}. Nous avons donc pu construire une séquence de longueur $m-1$ qui est plus courte que la séquence de longueur minimale, ce qui est absurde. On ne peut donc pas avoir dans ce cas plus d'une commande de la forme "supprimer r de $P[x,y]$".

(b) Le droit r apparaît dans une cellule (s,o) différente des cellules, (s_{n-q},o_{n-q}) et (s_n,o_n) :
Considérons la séquence

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \Delta_2 \ldots \longrightarrow^{\theta_{n-q-1}}_{\alpha_{n-q-1}} \Delta_{n-q-1} \longrightarrow^{\theta'_{n-q+1}}_{\alpha_{n-q+1}} \Delta'_{n-q+1}$$
$$\longrightarrow^{\theta'_{n-q+2}}_{\alpha_{n-q+2}} \Delta'_{n-q+2} \cdots \longrightarrow^{\theta'_{n-1}}_{\alpha_{n-1}} \Delta'_{n-1} \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} \Delta'_{n+1}$$
$$\longrightarrow^{\theta'_{n+2}}_{\alpha_{n+2}} \Delta'_{n+2} \ldots \longrightarrow^{\theta'_m}_{\alpha_m} \Delta'_m \quad (5)$$

dans laquelle pour tout entier $l \in \{n-q+1,\ldots,m\}$:
– $S'_l = S_l$,
– $O'_l = O_l$,
– $\theta'_l(x) = \theta_l(x)$ pour toute variable x de type sujte,
– $\theta'_l(y) = \theta_l(y)$ pour toute variable y type objet et

– $P'_l[s,o] = \begin{cases} P_l[s,o] \cup \{r\} & \text{si } (s,o) = (s_{n-q},o_{n-q}) \\ & \text{ou } (s,o) = (s_n,o_n) \\ P_l[s,o] & \text{sinon.} \end{cases}$

Cette séquence est plus courte que la séquence (1). Les commandes α_l, $l > n-q$ restent applicables. En effet, $P_l[s,o] \subseteq P'_l[s,o]$ pour tout entier $l \in \{0,\ldots,n-q-1,n-q+1,\ldots,m\}$, pour tout sujet $s \in S_l$ et pour tout objet $o \in O_l$. Si la commande α_m fait apparaître le droit r dans Δ_{m-1} alors elle fera également apparaître r dans Δ'_{m-1}. Nous avons donc montré dans les deux cas possibles pour l'apparition du droit r qu'on ne pouvait pas avoir, dans la séquence de longueur minimale, plus d'une commande de la forme "supprimer r de $P[x,y]$".

⊣

⊣

Lemme 11 *Soit* Π *un système de protection mono-opérationnel,* r *droit,* Δ *un état de protection et*

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \dots \longrightarrow^{\theta_{m-1}}_{\alpha_{m-1}} \Delta_{m-1} \longrightarrow^{\theta_m}_{\alpha_m} \Delta_m \textit{ (1)}$$

une séquence de longueur minimale telle que $\Delta_0 = \Delta$ et au terme de laquelle la commande α_m fait apparaître r dans Δ_{m-1}. Il existe au plus une commande $\alpha_l, l \in \{1, \dots, m-1\}$ de la forme "créer le sujet x ".

Preuve: Supposons qu'il existe deux commandes α_l et α_q de la forme "céer le sujet x" avec l'assignation $\theta_l(x) = s_l$ et $\theta_q(x) = s_q, l < q$ où $l, q \in \{1, \dots, m-1\}$. Soit la séquence obtenue en supprimant la commande α_q :

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \dots \longrightarrow^{\theta_{l-1}}_{\alpha_{l-1}} \Delta_{l-1} \longrightarrow^{\theta_l}_{\alpha_l} \Delta_l \longrightarrow^{\theta_{l+1}}_{\alpha_{l+1}} \dots \longrightarrow^{\theta_{q-1}}_{\alpha_{q-1}} \Delta_{q-1}$$

$$\longrightarrow^{\theta'_{q+1}}_{\alpha_{q+1}} \Delta'_{q+1} \longrightarrow^{\theta'_{q+2}}_{\alpha_{q+2}} \Delta'_{q+2} \dots \longrightarrow^{\theta'_m}_{\alpha_m} \Delta'_m \text{ (6)}$$

et dans laquelle pour tout entier $t \in \{q+1, \dots, m\}$:
– $S'_t = S_t \setminus \{s_q\}$,
– $O'_t = O_t \setminus \{s_q\}$,

– $\theta'_t(x) = \begin{cases} s_l & \text{si } \theta_t(x) = s_q \\ \theta_t(x) & \text{sinon,} \end{cases}$

– $\theta'_t(y) = \begin{cases} s_l & \text{si } \theta_t(y) = s_q \\ \theta_t(y) & \text{sinon,} \end{cases}$

– $P'_t[s,o] = \begin{cases} P_t[s,o] \cup P_t[s_q,o] & \text{si } s = s_l \text{ et } o \neq s_l \\ P_t[s,o] \cup P_t[s,s_q] & \text{si } s \neq s_l \text{ et } o = s_l \\ P_t[s,o] \cup P_t[s_q,o] \cup P_t[s,s_q] \cup P_t[s_q,s_q] & \text{si } s = s_l \text{ et } o = s_l \\ P_t[s,o] & \text{sinon.} \end{cases}$

Le lecteur vérifiera que les commandes et les assignations, de cette séquence, permettent de passer de Δ_0 à Δ'_m. Le droit r apparaît dans la séquence (6) dans la cellule où il apparaît dans la séquence de longueur minimale, sauf dans deux cas : le cas où le droit r apparaît dans la cellule $(s_q, o), o \neq s_q \in O_m$ et le cas où le droit r apparaît dans la cellule (s_q, s_q). Dans le premier cas le droit r apparaîtra dans la cellule (s_l, o) dans la séquence (6) et dans le deuxième cas il apparaîtra dans la cellule (s_l, s_l). Nous avons ainsi pu construire une séquence de longueur $m-1$ qui est plus courte que la séquence de longueur minimale. Nous avons donc une contradiction. Par conséquent on ne peut pas avoir, dans ce cas, plus d'une commande de la forme "créer le sujet x". $\dashv$

Lemme 12 *Soit Π un système de protection mono-opérationnel, r un droit, Δ un état de protection et*

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{m-1}}_{\alpha_{m-1}} \Delta_{m-1} \longrightarrow^{\theta_m}_{\alpha_m} \Delta_m \textit{ (1)}$$

une séquence de longueur minimale telle que $\Delta_0 = \Delta$ et au terme de laquelle la commande α_m fait apparaître r dans Δ_{m-1}. Il n'existe aucune commande $\alpha_l, l \in \{1, \ldots, m-1\}$ de la forme "créer l'objet y ".

Preuve: Sans perte de généralité, nous supposons que $|S_0| > 0$. Soit α_n la dernière commande de la forme "créer l'objet y" avec $\theta_n(y) = o_n$. Considérons la séquence :

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{n-1}}_{\alpha_{n-1}} \Delta_{n-1} \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} \Delta'_{n+1}$$

$$\longrightarrow^{\theta'_{n+2}}_{\alpha_{n+2}} \ldots \longrightarrow^{\theta'_m}_{\alpha_m} \Delta'_m \text{ (7)}$$

dans laquelle pour tout entier $l \in \{n+1, \ldots, m\}$:
- $S'_l = S_l$,
- $O'_l = O_l \setminus \{o_n\}$,
- $\theta'_l(x) = \theta_l(x)$ pour toute variable x de type sujet,
- $\theta'_l(y) \begin{cases} s_0 & \text{si } o = o_n \\ \theta'_l(y) & \text{sinon,} \end{cases}$
- $P'_l[s,o] = \begin{cases} P_l[s,o] \cup P_l[s,o_n] & \text{si } s = s_0 \\ P_l[s,o] & \text{sinon,} \end{cases}$

où $s_0 \in S_0$ est fixé. Le lecteur vérifiera que les commandes et les assignations de cette séquence, permettent de passer de Δ_0 à Δ'_m. Si le droit r apparaît dans la cellule $(s, o_n), s \in S_m$ alors il apparaîtra dans la cellule (s, s_0) dans la séquence (7). sinon l'apparition se fait dans la même cellule dans les séquences (1) et (7). Nous avons donc pu construire une séquence de longueur $m-1$ qui est plus courte que la séquence de longueur minimale, ce qui est absurde. On ne peut donc pas avoir une commande de la forme "créer l'objet y" dans la séquence de longueur minimale. $\dashv$

Nous pouvons maintenant, grâce aux lemmes précédents, donner la preuve du théorème suivant.

Théorème 10 *Il existe un algorithme, pour décider si un système de protection mono-opérationnel Π, par rapport à une configuration initial Δ, est sûr pour un droit r.*

Preuve: Les lemmes 1, 2, 3 et 4 nous permettent de conclure que la séquence de longueur minimale au terme de laquelle le droit r apparait, a les propriétés suivantes :
- le nombre de commandes de la forme "créer le sujet x" est au plus égal à 1,
- le nombre de commandes de la forme "supprimer r_l de $P[x,y]$", $l \in \{1, \ldots, m-1\}$ est au plus égal à 1 et
- le nombre de commandes de la forme "ajouter r_l dans $P[x,y]$", $r_l \neq r, l \in \{1, \ldots, m-1\}$ est au plus égal à $(|S_0|+1) \times (|O_0|+1) \times |R-1|$.

La longueur m de la séquence minimale est donc borné par $(|S_0| + 1) \times (|O_0| + 1) \times |R - 1| + 3$. Ce qui permet de construire un algorithme qui testera toute les combinaisons possibles d'au plus m commandes. $\dashv$
Le problème de la protection est donc décidable, pour la classe des systèmes de protection mono-opérationnels.

Annexe B

décidabilité du problème de la protection pour la classe des systèmes de protection monotones monoconditionnels

Dans cette annexe nous montrerons que le problème de la protection est décidable, pour la classe des systèmes de protection monotones monoconditionnels. Pour cela nous allons proposer quatre lemmes avec leur démonstration.

Nous rappelons que dans ce qui suit l'ensemble des sujets est disjoint de l'ensemble des objets. L'unique différence avec le cas où l'ensemble des sujets est inclu dans celui des objets est que l'opération primitive "créer le sujet x" n'ajoute pas d'élément dans l'ensemble d'objets courant. Même chose pour l'opération primitive "détruire le sujet x" elle ne supprime pas d'élément de l'ensemble d'objets courant.

Définition 2 *Soit* Π *un système de protection monotone monoconditionnel,* r *un droit et* Δ *un état de protection. On dira que la séquence*

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_n}_{\alpha_n} \Delta_n \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} \Delta_{n+1}$$

où $\Delta = \Delta_0$*, est une chaîne de* $r_1 \in (s_1, o_1)$*,* $s_1 \in S_1$*,* $o_1 \in O_1$ *à* $r_{n+1} \in (s_{n+1}, o_{n+1})$*,* $s_{n+1} \in S_{n+1}$*,* $o_{n+1} \in O_{n+1}$ *lorsque la condition suivantes est vérifiée pour tout entier* $l \in \{1, \ldots, n+1\}$ *:*

- *si "*$r_l \in P[x, y]$*" est la condition de de la commande* α_l *alors on a :*
  $r_l \in P_{l-1}[\theta_l(x), \theta_l(y)]$ *et (*$r_l \notin P_{l-2}[\theta_l(x), \theta_l(y)]$ *ou* $\theta_l(x) \notin S_{l-2}$ *ou* $\theta_l(y) \notin O_{l-2}$*) avec* $\theta_l(x) = s_l$ *et* $\theta_l(y) = o_l$*).*

Remarque :

1. Sans perte de généralité, nous supposons que $|S_0| \neq 0$ et $|O_0| \neq 0$.
2. Si $r' \notin P_l[s, o]$ alors pour tout entier $q \in \{0, ..., l-1\}$, si $s \in S_q$ et $o \in O_q$ alors $r' \notin P_q[s, o]$ si $s \in S_q$ et $o \in O_q$ pour tout entier $0 \leqslant q \leqslant l-1$, car le système de protection Π considéré est monotone.

Théorème 11 *Il existe un algorithme pour décider, si un système de protection monotone monoconditionnel* Π*, est sûr pour un droit* r*, relativement à un état initial* Δ*.*

Preuve: Considérons dabord le cas où aucune commande de la séquence qui fait apparaître r n'a sa condition qui porte sur une nouvelle cellule.

Lemme 13 *Soit* Π *un système de protection monotone monoconditionnel,* r *un droit,* Δ *un état deprotection et*

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \dots \longrightarrow^{\theta_n}_{\alpha_n} \Delta_n \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} \Delta_{n+1} \text{ (1)}$$

une séquence de longueur minimale telle que $\Delta_0 = \Delta$ *et au terme de laquelle* α_{n+1} *fait apparaître* r *dans* Δ_n*. Si toute commande* $\alpha_l, 1 \leq l \leq n$*, est telle que sa condition "si* $r_l \in P[x, y]$*" vérifie :*

- $\theta_l(x) \in S_0$ *et*
- $\theta_l(y) \in O_0$

alors $n \leq |S_0| \times |O_0| \times (|R| - 1) + 2$*.*

Preuve:
Soit (s_*, o_*) la cellule dans laquelle le droit r apparaît. On distingue deux cas :

1. $s_* \in S_0$ et $o_* \in O_0$,
2. $s_* \notin S_0$ ou $s_* \notin O_0$.

1. $s_* \in S_0$ et $o_* \in O_0$
   Dans ce cas on a les deux propriétés (a) et (b) suivantes :
   (a) Pour tout entier $l \in \{1, \dots, n\}$, le corps de la commande α_l contient une opération primitive de la forme "introduire r' dans $P[x, y]$", $r' \neq r$:

   Preuve: Soit α_q, une commande qui ne contient aucune opération primitive de la forme "introduire r' dans $P[x, y]$". Π est un système de protection monotone monoconditionnel, le corps de la commande α_q ne peut donc contenir que des opérations primitives de la forme "créer le sujet x" ou de la forme "créer l'objet y". Soit $s_1, \dots, s_m$ et $o_1, \dots, o_{m'}$ les sujets et les objets créés par la commande α_q.
   Considérons la séquence :

   $$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \dots \longrightarrow^{\theta_{q-1}}_{\alpha_{q-1}} \Delta_{q-1} \longrightarrow^{\theta'_{q+1}}_{\alpha_{q+1}} \Delta'_{q+1} \longrightarrow^{\theta'_{q+2}}_{\alpha_{q+2}} \Delta'_{q+2} \dots \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} \Delta'_{n+1} \text{ (2)}$$

   dans laquelle pour tout entier $l \in \{q + 1, \dots, n + 1\}$:

   - $S'_l = S_l \setminus \{s_1, \dots, s_m\}$,
   - $O'_l = O_l \setminus \{o_1, \dots, o_{m'}\}$,

– $\theta'_l(x) = \begin{cases} s_0 & \text{si } \theta_l(x) \in \{s_1, \ldots, s_m\} \\ \theta_l(x) & \text{sinon,} \end{cases}$

– $\theta'_l(y) = \begin{cases} o_0 & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'}\} \\ \theta_l(y) & \text{sinon,} \end{cases}$

– $P'_l[s,o] = \begin{cases} P_l[s,o] \bigcup \cup_{t=1}^{m} P_l[s_t,o] & \text{si } s = s_0 \text{ et } o \neq o_0 \\ P_l[s,o] \bigcup \cup_{t=1}^{m'} P_l[s,o_t] & \text{si } o = o_0 \text{ et } s \neq s_0 \\ P_l[s,o] \bigcup \cup_{k=1}^{m} \cup_{t=1}^{m'} P_l[s_k,o_t] & \text{si } o = o_0 \text{ et } s = s_0 \\ P_l[s,o] & \text{sinon,} \end{cases}$

où $s_0 \in S_0$ et $o_0 \in O_0$ sont fixés. Le lecteur vérifiera que les assignations et les commandes, de cette séquence, permettent de passer de Δ_0 à Δ'_{n+1}, la raison en est que $P_l[s,o] \subseteq P'_l[s,o]$ pour tout entier $l \in \{0, \ldots, q-1, q+1, \ldots n+1\}$, pour tout sujet $s \in S_l$ et pour tout objet $o \in O_l$. Si la commande α_{n+1} fait apparaître le droit r dans Δ_n alors elle le fera aussi apparaître dans Δ'_n. Nous avons pu construire une séquence de longueur inférieure à celle de la séquence minimale, ce qui est absurde. $\dashv$

Remarque : Nous notons par, "introduire r_1 dans $P[x_1, y_1]$", "introduire r_2 dans $P[x_2, y_2]$",..., "introduire r_{m_l} dans $P[x_{m_l}, y_{m_l}]$ ", les opérations primitives de la forme "introduire r' dans $p[x,y]$", de la commande α_l, $l \in \{1, \ldots, n\}$.

(b) Pour tout entier $l \in \{1, \ldots, n\}$, il existe un entier $t > l$ et un entier $f \in \{1, \ldots, m_l\}$ tel que la commande α_t a sa condition "$r_t \in P[x,y]$" qui vérifie $r_t = r_f$ et $(\theta_t(x), \theta_t(y)) = (\theta_l(x_f), \theta_l(y_f))$:

Preuve:
Soit une commande α_q, $q \in \{1, \ldots, n\}$ tel que pour tout entier $t > q$ et pour tout entier $f \in \{1, \ldots m_q\}$, la commande α_t a sa condition "$r_t \in P[x,y]$", et $r_t \neq r_f$ ou $(\theta_t(x), \theta_t(y)) \neq (\theta_q(x_f), \theta_q(o_f))$. Comme précédemment considérons la séquence :

$$\Delta_0 \longrightarrow_{\alpha_1}^{\theta_1} \Delta_1 \longrightarrow_{\alpha_2}^{\theta_2} \ldots \longrightarrow_{\alpha_{q-1}}^{\theta_{q-1}} \Delta_{q-1} \longrightarrow_{\alpha_{q+1}}^{\theta'_{q+1}} \Delta'_{q+1} \longrightarrow_{\alpha_{q+2}}^{\theta'_{q+2}}$$
$$\Delta'_{q+2} \ldots \longrightarrow_{\alpha_{n+1}}^{\theta'_{n+1}} \Delta'_{n+1} \quad (3)$$

dans laquelle pour tout entier $l \in \{q+1, \ldots, n+1\}$:

– $S'_l = S_l \setminus \{s_1, \ldots, s_m\}$,
– $O'_l = O_l \setminus \{o_1, \ldots, o_{m'}\}$,

– $\theta'_l(x) = \begin{cases} s_0 & \text{si } \theta_l(x) \in \{s_1, \ldots, s_m\} \\ \theta_l(x) & \text{sinon,} \end{cases}$

– $\theta'_l(y) = \begin{cases} o_0 & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'}\} \\ \theta_l(y) & \text{sinon,} \end{cases}$

où $s_0 \in S_0$, $o \in O_0$ sont fixés et où $s_1, \ldots, s_m$ et $o_1, \ldots, o_{m'}$ sont les sujets et les objets créés par la commande α_q. Nous avons ainsi pu construire une séquence de longueur inférieure à celle de la séquence de longueur minimale, ce qui est absurde. De ce fait toutes les commandes α_l de la séquence de longueur minimale contiennent des opérations primitives de la forme "introduire r' dans $P[x, y]$", $r' \neq r$, telles qu'il existe $t > l$ qui a sa condition de la forme "$r_t \in P[x, y]$", où $r_t = r'$ et $(\theta_t(x), \theta_t(y)) = (\theta_l(x), \theta_l(y))$. Nous savons aussi par hypothèse que toute commande $\alpha_l, 1 \leq l \leq n$ a sa condition "$r_l \in P[x, y]$" avec $\theta_l(x) \in S_0, \theta_l(y) \in O_0$. Le nombre de commandes dans la séquence (1) qui permet d'arriver à l'apparition est donc au plus égal à $|S_0| \times |O_0| \times (|R| - 1)$+1. ⊣

2. $s_* \notin S_0$ ou $o_* \notin O_0$:

   Notons qu'on a besoin d'au plus deux commandes pour créer une nouvelle cellule, considérons le cas où $s_* \notin S_0$ et $o_* \notin O_0$ (les autres cas se traiteraient de la même façon). Soit $\alpha_b, 1 \leq b \leq n + 1$, la commande qui contient l'opération primitive "créer le sujet x" et qui vérifie $\theta_b(x) = s_*$ et soit $\alpha_c, 1 \leq c \leq n + 1$, celle qui contient l'opération primitive "créer l'objet y" et qui vérifie $\theta_c(y) = o_*$. Sans perte de généralité, nous considérons que $b < c$.
   Il est facile de montrer, comme précédement, que toutes les commandes α_l $l \neq b$ et de $l \neq c$ de la séquence minimale contiennent des opérations primitives de la forme "introduire r' dans $P[x, y]$", $r' \neq r$, telles qu'il existe $t > l$ qui a sa condition de la forme "$r_t \in P[x, y]$", où $r_t = r'$ et où $(\theta_t(x), \theta_t(y)) = (\theta_l(x), \theta_l(y))$. Nous savons aussi par hypothèse que toute commande $\alpha_l, 1 \leq l \leq n$, a sa condition "$r_l \in P[x, y]$" avec $\theta_l(x) \in S_0, \theta_l(y) \in O_0$. Le nombre de commandes différentes de α_b et α_c est au plus égal à $|S_0| \times |O_0| \times (|R| - 1)$, on peut donc borner le nombre n de commandes par $|S_0| \times |O_0| \times (|R| - 1)$+3.

⊣

Nous pouvons à présent considérer le cas plus général dans lequel il existe dans la séquence au terme de laquelle le droit r apparaît des conditions qui portent sur de nouvelles cellules.

Lemme 14 *Soit Π un système de protection monotone monoconditionnel, r un droit, Δ un état de protection et*

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_n}_{\alpha_n} \Delta_n \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} \Delta_{n+1} \textit{ (4)}$$

une séquence de longueur minimale telle que $\Delta_0 = \Delta$ et au terme de laquelle la commande α_{n+1} fait apparaître r dans Δ_n. Soit $d \geq 0$ le nombre de commandes dans $\{\alpha_1, \ldots, \alpha_{n+1}\}$ dont le test porte sur une cellule qui n'est pas initialement dans Δ_0. On a : $n \leq |S_0| \times |O_0| \times (|R - 1|) + 3d + 2$.

Preuve: Soit $\alpha_{k_1}, \alpha_{k_2}, ..., \alpha_{k_d}$, où $1 \leq k_1$ et $k_d \leq n + 1$, des commandes avec leur condition de la forme "$r_{k_l} \in P[x, y]$" telles que $\theta_{k_l}(x) \notin S_0$ ou $\theta_{k_l}(y) \notin O_0$. Pour tout entier $l \in \{1, \ldots, d\}$ on a :

- si $\theta_{k_l}(x) \notin S_0$ alors soit α_{m_l} est la commande qui contient l'opération primitive "créer le sujet x" avec $\theta_{m_l}(x) = \theta_{k_l}(x)$,
- si $\theta_{k_l}(y) \notin O_0$ alors soit $\alpha_{m'_l}$ est la commande qui contient l'opération primitive "créer l'objet y" avec $\theta_{m'_l}(y) = \theta_{k_l}(y)$,
- soit $\alpha_{m''_l}$ la commande qui contient l'opération primitive "introduire r_{k_l} dans $P[x, y]$" avec $\theta_{m''_l}(x) = \theta_{k_l}(x), \theta_{m''_l}(y) = \theta_{k_l}(y)$.

Notons que $m_l, m'_l, m''_l \in \{1, ..., k_l - 1\}$. Sans perte de généralité, nous supposons que $m_l \leq m'_l \leq m''_l$.

Remarque : Dans ce qui suit, $\theta_{k_1}(x)$ sera noté s_{k_1} et $\theta_{k_1}(y)$ sera noté o_{k_1}.

Soit (s_*, o_*), la cellule dans laquelle le droit r apparaît. On a besoin d'au plus deux commandes pour créer une nouvelle cellule. Sans perte de généralité, considérons dans cette démonstration le cas où $s_* \notin S_0$ et $o_* \notin O_0$. Soit $\alpha_b, 1 \leq b \leq n + 1$, la commande qui contient l'opération primitive "créer le sujet x" avec $\theta_b(x) = s_*$ et $\alpha_c, 1 \leq c \leq n+1$, celle qui contient l'opération primitive "créer l'objet y" avec $\theta_c(y) = o_*$. Sans perte de généralité, nous considèrons que $b < c$.

Démontrons le fait suivant :

Fait : Pour tout entier $l \notin \{m_1, \ldots, m_d, m'_1, \ldots, m'_d, m''_1, \ldots, m''_d, b, c, n + 1\}$, les commandes α_l ont des opérations primitives de la forme "introduire r' dans $P[x, y]$", $r' \neq r$, pour lesquelles il existe $t > l$ tel que la commande α_t a sa condition de la forme ' $r_t \in P[x, y]$", où $r_t = r'$, $(\theta_t(x), \theta_t(y)) = (\theta_l(x), \theta_l(y))$, $\theta_l(x) \in S_0$ et $\theta_l(y) \in O_0$.

Si la séquence minimale contient une commande α_q qui ne satisfait pas cette propriété alors on considère la séquence :

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_{q-1}}_{\alpha_{q-1}} \Delta_{q-1} \longrightarrow^{\theta'_{q+1}}_{\alpha_{q+1}} \Delta'_{q+1} \longrightarrow^{\theta'_{q+2}}_{\alpha_{q+2}} \Delta'_{q+2} \ldots \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} \Delta'_{n+1} \quad (5)$$

dans laquelle pour tout entier $l \in \{q + 1, \ldots, n + 1\}$ on a :

- $S'_l = S_l \setminus \{s_1, \ldots, s_m\}$,
- $O'_l = O_l \setminus \{o_1, \ldots, o_{m'}\}$,
- $\theta'_l(x) = \begin{cases} s_0 & \text{si } \theta_l(x) \in \{s_1, \ldots, s_m\} \\ \theta_l(x) & \text{sinon,} \end{cases}$
- $\theta'_l(y) = \begin{cases} o_0 & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'}\} \\ \theta_l(y) & \text{sinon,} \end{cases}$

où $s_0 \in S_0$ et $o_0 \in O_0$ sont fixés et où $s_1, \ldots, s_m$ et $o_1, \ldots, o_{m'}$ sont les sujets et les objets créés par la commande α_q.

Nous avons donc pu trouver, une séquence plus courte que la séquence minimale. Notons que, dans cette séquence (5), la commande α_{n+1} fait apparaître le droit r dans Δ_n. Nous avons donc pu trouver une séquence, au terme de laquelle le droit r apparaît, de longueur inférieur à celle de la séquence de longueur minimale. On a donc une contradiction. La séquence minimale contient ainsi au plus $|S_0| \times |O_0| \times (|R| - 1) + 3d + 3$ commandes. Ceci termine la preuve du lemme 14. $\dashv$

Il suffira donc de borner d pour borner la séquence minimale. Pour ce faire, nous allons utiliser les résultats des lemmes 15 et 16 suivants.

Lemme 15 *Soit Π un système de protection monotone monoconditionnel, r un droit, Δ un état de protection et*

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \dots \longrightarrow^{\theta_n}_{\alpha_n} \Delta_n \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} \Delta_{n+1} \textit{ (4)}$$

*une séquence de longueur minimale telle que $\Delta_0 = \Delta$ et au terme de laquelle la commande α_{n+1} fait apparaître r dans Δ_n. Soit α_{k_1},$k_1 \in \{1, \dots, n+1\}$ la première commande qui a sa condition de la forme "$r_{k_1} \in P[x, y]$" avec $\theta_{k_1}(x) \notin S_0$ ou $\theta_{k_1}(y) \notin O_0$. Soit $\alpha_{m_1''}$ la commande qui contient l'opération primitive "*introduire r_{k_1} dans $P[x, y]$*" qui vérifie $\theta_{m_1''}(x) = \theta_{k_1}(x)$ et $\theta_{m_1''}(y) = \theta_{k_1}(y)$.*

On a :
– *une chaîne de $r_{m_1''+2} \in (s_{m_1''+2}, o_{m_1''+2})$ à $r_{n+1} \in (s_{n+1}, o_{n+1})$.*

Preuve:
Raisonnant par l'absurde, soit $t \in \{m_1'' + 2, \dots, n+1\}$ le plus grand entier tel que, α_t a sa condition "$r_t \in P[x, y]$" et est tel qu'il existe $q < t - 1$ tel que $r_t \in p_q[\theta_t(x), \theta_t(x)]$. Soit q le plus petit entier qui satisfait ce critère. Il existe deux cas possibles pour q : $q < m_1''$ et $q \geq m_1''$.

1. $q < m_1''$: Soit la séquence minimale

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \dots \longrightarrow^{\theta_q}_{\alpha_q} \Delta_q \longrightarrow^{\theta_{q+1}}_{\alpha_{q+1}} \dots \longrightarrow^{\theta_{m_1''}}_{\alpha_{m_1''}} \Delta_{m_1''} \longrightarrow^{\theta_{m_1''+1}}_{\alpha_{m_1''+1}} \dots \longrightarrow^{\theta_{t-1}}_{\alpha_{t-1}}$$
$$\Delta_{t-1} \longrightarrow^{\theta_t}_{\alpha_t} \Delta_t \longrightarrow^{\theta_{t+1}}_{\alpha_{t+1}} \dots \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} \Delta_{n+1} \text{ (6)}$$

La commande α_{n+1} qui fait apparaître r dans Δ_{n+1} contient nécessairement l'opération primitive "introduire r dans $P[x, y]$" avec $\theta(x) = s_*$ et $\theta(y) = o_*$, On distingue quatre cas :

– Cas 1 : La commande α_{n+1} contient l'opération primitive "créer le sujet x" et ne contient pas l'opération primitive "créer l'objet y".
– Cas2 : La commande α_{n+1} contient l'opération primitive "créer le sujet x" et l'opération primitive "créer l'objet y".
– Cas 3 : La commande α_{n+1} ne contient pas l'opération primitive "créer le sujet x" et contient l'opération primitive "créer l'objet y".

– Cas 4 : La commande α_{n+1} ne contient ni l'opération primitive "créer le sujet x" ni l'opération primitive "créer l'objet y".

Considérons la séquence :

$$\Delta_0 \longrightarrow_{\alpha_1}^{\theta_1} \ldots \longrightarrow_{\alpha_q}^{\theta_q} \Delta_q \longrightarrow_{\alpha_{q+1}}^{\theta_{q+1}} \ldots \longrightarrow_{\alpha_{m_1''}}^{\theta_{m_1''}} \Delta_{m_1''} \longrightarrow_{\alpha_t}^{\theta'_t} \Delta'_t \longrightarrow_{\alpha_{t+1}}^{\theta'_{t+1}} \ldots \longrightarrow_{\alpha_{n+1}}^{\theta'_{n+1}} \Delta'_{n+1} \quad (7)$$

dans laquelle pour tout entier $l \in \{t, \ldots, n+1\}$:

– $S'_l = S_l \setminus \{s_1, \ldots, s_m\}$,
– $O'_l = O_l \setminus \{o_1, \ldots, o_{m'}\}$,

et dans laquelle :

(a) si nous sommes dans le cas 1 alors :

– $\theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1, \ldots, s_m\} \\ \theta_l(x) & \text{sinon,} \end{cases}$
– $\theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'+1}\} \\ \theta_l(y) & \text{sinon,} \end{cases}$

(b) si nous sommes dans le cas 2 alors :

– $\theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1, \ldots, s_m\} \\ \theta_l(x) & \text{sinon,} \end{cases}$
– $\theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'}\} \\ \theta_l(y) & \text{sinon,} \end{cases}$

(c) si nous sommes dans le cas 3 alors :

– $\theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1, \ldots, s_{m+1}\} \\ \theta_l(x) & \text{sinon,} \end{cases}$
– $\theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'}\} \\ \theta_l(y) & \text{sinon,} \end{cases}$

(d) si nous sommes dans le cas 4 alors :

– $\theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1, \ldots, s_{m+1}\} \\ \theta_l(x) & \text{sinon,} \end{cases}$
– $\theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'+1}\} \\ \theta_l(y) & \text{sinon,} \end{cases}$

où $s_1, \ldots, s_m$ et $o_1, \ldots, o_{m'}$ sont les sujets et les objets créés par les commandes supprimées α_h, $m_1'' + 1 \leq h \leq t-1$, et où $(s_{m+1}, o_{m'+1}) = (s_*, o_*)$ est la cellule

dans laquelle le droit r apparaît dans la séquence de longueur minimale. Les assignations et les commandes permettent, dans les quatre cas, de passer de Δ_0 à Δ'_{n+1}. Dans les quatre cas possibles pour la commande α_n, si le droit r apparaît dans Δ_n alors il apparaîtra dans Δ'_n. Plus précisement, dans la cellule (s_*, o_{k_1}) dans le cas 1, dans la cellule (s_*, o_*) dans le cas 2, dans la cellule (s_{k_1}, o_*) dans la cas 3 et dans la cellule (s_{k_1}, o_{k_1}) dans le cas 4. Nous avons donc pu construire une séquence qui a un nombre de commandes inférieur à celui de la séquence de longueur minimale. Ce qui est absurde.

2. $q \geq m''_1$: Soit la séquence minimale

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \ldots \longrightarrow^{\theta_{m''_1}}_{\alpha_{m''_1}} \Delta_{m''_1} \longrightarrow^{\theta_{m''_1+1}}_{\alpha_{m''_1+1}} \ldots \longrightarrow^{\theta_q}_{\alpha_q} \Delta_q \longrightarrow^{\theta_{q+1}}_{\alpha_{q+1}} \ldots \longrightarrow^{\theta_{t-1}}_{\alpha_{t-1}}$$
$$\Delta_{t-1} \longrightarrow^{\theta_t}_{\alpha_t} \Delta_t \longrightarrow^{\theta_{t+1}}_{\alpha_{t+1}} \ldots \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} \Delta_{n+1} \text{ (8)}$$

Nous supposons comme dans (a), qu'il existe quatre cas possibles pour la commande α_{n+1}. Considérons la séquence :

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \ldots \longrightarrow^{\theta_{m''_1}}_{\alpha_{m''_1}} \Delta_{m''_1} \longrightarrow^{\theta_{m''_1+1}}_{\alpha_{m''_1+1}} \ldots \longrightarrow^{\theta_q}_{\alpha_q} \Delta_q \longrightarrow^{\theta'_t}_{\alpha_t} \Delta'_t \longrightarrow^{\theta'_{t+1}}_{\alpha_{t+1}}$$
$$\ldots \longrightarrow^{\theta'_{n+1}}_{\alpha_{n+1}} \Delta'_{n+1} \text{ (9)}$$

dans laquelle pour tout entier $l \in \{t, \ldots, n+1\}$, la construction des S'_l, O'_l, $\theta_l(x)$ et $\theta_l(y)$ se fait exactement comme dans (a) sauf que $s_1, \ldots, s_m$ et $o_1, \ldots, o_{m'}$ vont repésenter les sujets et les objets créés par les commandes supprimées α_h, $q+1 \leq h \leq t-1$. Les assignations et les commandes permettent de passer de Δ_0 à Δ'_{n+1}. Comme dans (a), dans les quatre cas possibles pour la commande α_{n+1}, si la commande α_{n+1} fait apparaître le droit r dans Δ_n alors elle le fera apparaître dans Δ'_n. Plus précisement, dans la cellule (s_*, o_{k_1}) dans le cas 1, dans la cellule (s_*, o_*) dans le cas 2, dans la cellule (s_{k_1}, o_*) dans la cas 3 et dans la cellule (s_{k_1}, o_{k_1}) dans le cas 4. Nous avons ainsi pu construire une séquence qui a un nombre de commandes inférieur à celui de la séquence de longueur minimale, ce qui est absurde.

On a donc une chaîne de $r_{m''_1+2} \in (s_{m''_1+2}, o_{m1''_1+2})$ à $r_{n+1} \in (s_{n+1}, o_{n+1})$.

Ceci termine la preuve du lemme 15 $\dashv$

Nous allons utiliser le résulta du lemme 15 pour démontrer le lemme 16 ci-dessous. Ce qui nous permettra de borner d et par conséquent de borner la longueur de la séquence minimale, au terme de laquelle le droit r apparaît.

Lemme 16 *Soit Π un système de protection monotone monoconditionnel, r un droit, Δ un état de protection et*

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_n}_{\alpha_n} \Delta_n \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} \Delta_{n+1} \textit{ (4)}$$

une séquence de longueur minimale telle que $\Delta_0 = \Delta$ et au terme de laquelle la commande α_{n+1} fait apparaître r dans Δ_n. Soit $\alpha_{k_1}, \alpha_{k_2}, ..., \alpha_{k_d}$, où $1 \leq k_1$ et $k_d \leq n+1$, des commandes avec leurs conditions de la forme "$r_{k_l} \in P[x,y]$" telles que $\theta_{k_l}(x) \notin S_0$ ou $\theta_{k_l}(y) \notin O_0$. On a :

- *$\forall l \neq q$, $r_{k_l} \neq r_{k_q}$ où $l, q \in \{1, \ldots, d\}$*

Preuve:

Soit α_{k_1},$k_1 \in \{1, \ldots, n+1\}$, la première commande qui a sa condition de la forme "$r_{k_1} \in p[x,y]$" telle que $\theta_{k_1}(x) \notin S_0$ ou $\theta_{k_1}(y) \notin O_0$. Soit $\alpha_{m''_1}$ la commande qui contient l'opération primitive "introduire r_{k_1} dans $P[x,y]$" telle que $\theta_{m''_1}(x) = \theta_{k_1}(x)$ et $\theta_{m''_1}(y) = \theta_{k_1}(y)$. Supposons qu'il existe deux commandes α_{k_t} et α_{k_q} dont les conditions sont respectivement "$r_{k_t} \in p[x,y]$" et "$r_{k_q} \in p[x,y]$" avec :

- $\theta_{k_t}(x) \notin S_0$ ou $\theta_{k_t}(y) \notin O_0$ et
- $\theta_{k_q}(x) \notin S_0$ ou $\theta_{k_q}(y) \notin O_0$.

Supposons aussi que $r_{k_t} = r_{k_q}$

Remarque : Nous noterons respectivement $\theta_{k_t}(x), \theta_{k_q}(x), \theta_{k_t}(y)$ et $\theta_{k_q}(y)$ par s_{k_t}, s_{k_q}, o_{k_t} et o_{k_q}.

Soit la séquence minimale :

$$\Delta_0 \underset{\alpha_1}{\overset{\theta_1}{\longrightarrow}} \ldots \underset{\alpha_{m''_1}}{\overset{\theta_{m''_1}}{\longrightarrow}} \Delta_{m''_1} \underset{\alpha_{m''_1+1}}{\overset{\theta_{m''_1+1}}{\longrightarrow}} \ldots \underset{\alpha_{k_t}}{\overset{\theta_{k_t}}{\longrightarrow}} \Delta_{k_t} \underset{\alpha_{k_t+1}}{\overset{\theta_{k_t+1}}{\longrightarrow}} \ldots \underset{\alpha_{k_q}}{\overset{\theta_{k_q}}{\longrightarrow}} \Delta_{k_q} \underset{\alpha_{k_q+1}}{\overset{\theta_{k_q+1}}{\longrightarrow}} \ldots \underset{\alpha_{n+1}}{\overset{\theta_{n+1}}{\longrightarrow}} \Delta_{n+1} \quad (10)$$

Nous supposons comme précédement, qu'il existe quatre cas possibles pour la commande α_{n+1}. Considérons la séquence :

$$\Delta_0 \underset{\alpha_1}{\overset{\theta_1}{\longrightarrow}} \ldots \underset{\alpha_{m''_1}}{\overset{\theta_{m''_1}}{\longrightarrow}} \Delta_{m''_1} \underset{\alpha_{m''_1+1}}{\overset{\theta_{m''_1+1}}{\longrightarrow}} \ldots \underset{\alpha_{k_t-1}}{\overset{\theta_{k_t-1}}{\longrightarrow}} \Delta_{k_t-1} \underset{\alpha_{k_q}}{\overset{\theta'_{k_q}}{\longrightarrow}} \Delta'_{k_q} \underset{\alpha_{k_q+1}}{\overset{\theta'_{k_q+1}}{\longrightarrow}} \ldots \underset{\alpha_{n+1}}{\overset{\theta'_{n+1}}{\longrightarrow}} \Delta'_{n+1} \quad (11)$$

dans laquelle pour tout entier $l \in \{k_q, \ldots, n+1\}$:

- $S'_l = S_l \setminus \{s_1 \ldots s_m\}$,
- $O'_l = O_l \setminus \{o_1 \ldots o_{m'}\}$,

et dans laquelle :

1. si nous sommes dans le cas 1 alors :

   - $\theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1, \ldots, s_m\} \setminus \{s_{k_q}\} \\ s_{k_t} & \text{si } \theta_l(x) = s_{k_q} \\ \theta_l(x) & \text{sinon,} \end{cases}$

– $\theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'+1}\} \setminus \{o_{k_q}\} \\ o_{k_l} & \text{si } \theta_l(y) = o_{k_q} \\ \theta_l(y) & \text{sinon,} \end{cases}$

2. si nous sommes dans le cas 2 alors :

– $\theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1, \ldots, s_m\} \setminus \{s_{k_q}\} \\ s_{k_t} & \text{si } \theta_l(x) = s_{k_q} \\ \theta_l(x) & \text{sinon,} \end{cases}$

– $\theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'}\} \setminus \{o_{k_q}\} \\ o_{k_l} & \text{si } \theta_l(y) = o_{k_q} \\ \theta_l(y) & \text{sinon,} \end{cases}$

3. si nous sommes dans le cas 3 alors :

– $\theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1, \ldots, s_{m+1}\} \setminus \{s_{k_q}\} \\ s_{k_t} & \text{si } \theta_l(x) = s_{k_q} \\ \theta_l(x) & \text{sinon,} \end{cases}$

– $\theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'}\} \setminus \{o_{k_q}\} \\ o_{k_l} & \text{si } \theta_l(y) = o_{k_q} \\ \theta_l(y) & \text{sinon,} \end{cases}$

4. si nous sommes dans le cas 4 alors :

– $\theta'_l(x) = \begin{cases} s_{k_1} & \text{si } \theta_l(x) \in \{s_1, \ldots, s_{m+1}\} \setminus \{s_{k_q}\} \\ s_{k_t} & \text{si } \theta_l(x) = s_{k_q} \\ \theta_l(x) & \text{sinon,} \end{cases}$

– $\theta'_l(y) = \begin{cases} o_{k_1} & \text{si } \theta_l(y) \in \{o_1, \ldots, o_{m'+1}\} \setminus \{o_{k_q}\} \\ o_{k_l} & \text{si } \theta_l(y) = o_{k_q} \\ \theta_l(y) & \text{sinon,} \end{cases}$

où $s_1, \ldots, s_m$ et $o_1, \ldots, o_{m'}$ sont les sujets et les objets créés par les commandes supprimées $\alpha_h, k_t \leq h \leq k_{q-1}$ et où $(s_{m+1}, o_{m'+1}) = (s_*, o_*)$ est la cellule dans laquelle le droit r apparaît dans la séquence de longueur minimale. Les assignations et les commandes permettent de passer de Δ_0 à Δ'_{n+1}, grâce au fait d'avoir une chaîne de $r_{m''_1+2} \in (s_{m''_1+2}, o_{m''_1+2})$ à $r_{n+1} \in (s_{n+1}, o_{n+1})$. On peut facilement vérifier, dans les quatre cas possibles pour la commande α_{n+1}, que si r apparaît dans Δ_n alors il apparaîtra dans Δ'_n. Plus précisement, dans la cellule (s_*, o_{k_1}) dans le cas 1, dans la cellule (s_*, o_*) dans le cas 2, dans la cellule (s_{k_1}, o_*) dans la cas 3 et dans la cellule (s_{k_1}, o_{k_1}) dans le cas 4. Nous avons pu construire une séquence ayant un nombre de commandes iférieur à celui de la séquence (4), ce qui est absurde. On ne peut donc pas avoir deux commandes qui testent le même droit dans des nouvelles cellules. Ceci termine la preuve du lemme 16 $\dashv$

De ce fait, avec les lemmes 15 et 16, on en conclut que si

$$\Delta_0 \longrightarrow^{\theta_1}_{\alpha_1} \Delta_1 \longrightarrow^{\theta_2}_{\alpha_2} \ldots \longrightarrow^{\theta_n}_{\alpha_n} \Delta_n \longrightarrow^{\theta_{n+1}}_{\alpha_{n+1}} \Delta_{n+1}$$

est une séquence de longueur miniamle au terme de laquelle la commande α_{n+1} fait apparaître le droit r alors $n \leq (|S_0| \times |O_0| + 3) \times (|R| - 1) + 2$.

Le fait de borner le nombre de commandes nous permet de construire un algorithme qui testera toutes les combinaisons possible d'au plus $n + 1$ commandes, pour savoir si le droit r apparaît ou pas. Ainsi on pourra savoir si le système de proitection Π est sûr pour le droit r relativement à l'état de protection Δ. Ceci termine la preuve du théorème 11 $\dashv$

Nous avons donc pu montrer que le problème de la protection est décidable, pour la classe des systèmes de protection monotones monoconditionnels.