Université Pierre et Marie Curie

Rapport de stage de Master Sciences et Technologies
Mention Informatique
Spécialité Science et Technologie du Logiciel
Parcours Logiciels Sûrs

Formalisation, comparaison et implantation d'un modèle de contrôle d'accès à base de rôles

Lionel Habib

Septembre 2007

Encadrants : Mathieu Jaume
Charles Morisset

Table des matières

Remerciements

Je remercie Michèle Soria, responsable du Master Science et Technologie du Logiciel, pour m'avoir permis de suivre cette formation. Je remercie Mathieu Jaume, responsable du Parcours Logiciels Sûrs, pour son encadrement de qualité, Charles Morisset, doctorant dans l'équipe SPI, pour son aide précieuse dans la réalisation de ce travail et Thérèse Hardin, responsable de l'équipe SPI, pour son accueil au sein de l'équipe et sa grande disponibilité.

Je remercie également les membres de l'action ANR SSURF (*Safety and Security UndeR Focal*), dans laquelle mon stage s'est intégré, pour les interactions enrichissantes que nous avons eues.

Je remercie aussi Philippe Ayrault, Julien Blond, Richard Bonichon, Florian Brecht, Eric Jaeger, Alexandre Lejeune, Yvan Noyer, François Pessaux et Renaud Rioboo pour avoir partagé mon quotidien dans l'équipe SPI et avec qui j'ai eu de nombreuses discussions durant mon stage.

Enfin, je remercie Damien Doligez et Pierre Weis pour leurs conseils avisés dans l'utilisation de l'atelier Focal.

Résumé

Un des aspects de la sécurité en informatique concerne le contrôle des accès aux données d'un système pour lequel différentes politiques de sécurité peuvent être mises en application. Toutefois, rien ne sert de mettre en place une politique de sécurité pour gérer un système si les programmes chargés de garantir le bon fonctionnement de cette politique ne sont pas fiables. Ne pas apporter de garanties fortes sur la correction de tels programmes reviendrait à construire un château fort avec une porte en papier. Le travail présenté dans ce document adopte donc une approche formelle et a pour objectif d'obtenir ainsi une spécification et une implantation certifiée d'un modèle de contrôle d'accès à base de rôles. Un tel modèle est réputé très général et nous l'implantons dans l'atelier Focal. Nous fournissons d'autre part un schéma de traduction des modèles à base de treillis (plus anciens) vers les modèles à base de rôles. Un tel schéma de traduction permet de réutiliser l'implantation d'un modèle à base de rôles pour simuler un modèle à base de treillis.

1 Introduction

La protection des informations d'un système informatique est une préoccupation majeure. En effet, lorsqu'il s'agit de systèmes logiciels critiques, les propriétés de sécurité d'un système peuvent être vitales. Aussi, les Critères Communs [**?**] (recueil de normes définies par des agences gouvernementales) fournissent une méthodologie permettant d'atteindre des hauts niveaux de sécurité. Ils définissent à la fois un cadre de travail pour la conception et la réalisation de logiciels et une référence pour les utilisateurs de ces logiciels. Les hauts niveaux de sûreté des Critères Communs requièrent l'utilisation de méthodes formelles dans les étapes de spécification et de conception du logiciel. Selon les Critères Communs, un système est vu comme une installation donnée de technologies de l'information, avec un objectif et un environnement opérationnel particuliers. Dans le travail présenté, nous utilisons les méthodes formelles pour étudier certaines des propriétés classiques de sécurité des systèmes informatiques. Nous nous intéressons plus particulièrement au contrôle d'accès.

Le contrôle d'accès est un élément fondamental de la sécurité informatique. Il désigne tout mécanisme par lequel un système accorde ou révoque le droit à des entités actives d'accéder à des entités passives. Afin de régir les accès dans un système, une politique de sécurité, décrivant les propriétés que doivent vérifier les systèmes sûrs, doit être définie. Un moniteur de référence qui gére les changements d'états d'un système de contrôle d'accès en respectant une politique de sécurité peut être alors défini. Citons, par exemple, deux propriétés pouvant être décrites par une politique de sécurité. Le principe du moindre privilège (*least privilege*) caractérise le fait qu'un utilisateur doit avoir uniquement accès aux fonctionnalités dont il a besoin pour effectuer une action. Par exemple, cela consiste pour un administrateur à ne pas utiliser en permanence le compte *root*. La propriété de séparation des tâches (*separation of duties*), quant à elle, consiste à empêcher une même personne d'effectuer un ensemble d'opérations jugées incompatibles. Par exemple, on peut vouloir empêcher qu'une personne autorise un paiement dont elle a fait la demande. Au sein du système où est mis en place le contrôle d'accès, les sujets forment les entités actives, un sujet pouvant être la vision informatique de l'utilisateur, personne physique, et les objets caractérisent les entités passives. De plus, des permissions, spécifiant quelles opérations sont autorisées sur quels objets, interviennent dans les définitions des propriétés décrites par la politique de sécurité.

Deux formes de contrôle d'accès sont généralement considérés. Le premier est nommé contrôle d'accès discrétionnaire (DAC pour *Discretionary Access Control*) dans lequel le propriétaire d'une ressource fixe lui-même les droits d'accès de tous les utilisateurs du système sur celle-ci. L'adjectif discrétionnaire indique que la gestion des droits est laissée à la discrétion de l'utilisateur. Ce type d'accès est très largement utilisé, par exemple, dans les

systèmes d'exploitation à la UNIX. Le second est le contrôle d'accès obligatoire (MAC pour *Mandatory Access Control*) dans lequel le système gère directement les droits des utilisateurs. Ce contrôle est réalisé à l'aide d'informations de sécurité attribuées aux utilisateurs et aux objets. Un tel procédé est utilisé dans les systèmes militaires ou dans les grandes entreprises où les données manipulées peuvent être sensibles et/ou ne pas appartenir aux utilisateurs. Les deux principaux modèles de MAC sont ceux de Bell et LaPadula [**?**, **?**] et de la Muraille de Chine [**?**].

La suite de ce document porte sur le contrôle d'accès à base de rôles (RBAC pour *Role Based Access Control*). Une première tentative de formalisation d'un modèle RBAC basique a été introduite dans [**?**]. Cette présentation, connue sous le nom de RBAC92, a pour but d'identifier les concepts fondamentaux, principalement liés à la notion de rôles, qui définissent un modèle RBAC. Dans un second temps, le modèle RBAC96, défini dans [**?**], ajoute un certain nombre de notions comme celles d'utilisateurs ou de hiérarchie de rôles. C'est ce modèle que nous allons plus particulièrement étudier dans les sections suivantes. Enfin, ces modèles ont abouti à une standardisation exposée dans [**?**]. De plus, [**?**] étudie en détail le modèle RBAC. Bien que le modèle RBAC fût tout d'abord identifié comme faisant partie de la famille MAC, il a été montré qu'il pouvait aussi bien mettre en œuvre des politiques des modèles DAC [**?**, **?**] que MAC [**?**, **?**]. En effet, grâce à sa notion de rôles, RBAC possède un certain nombre d'avantages par rapport aux autres modèles de contrôle d'accès. Par exemple, le modèle RBAC permet une grande flexibilité étant donné qu'il est associé à une politique de sécurité très générique. Il peut être adapté afin de prévenir prévenir un flot d'information « descendant » comme le réalise le modèle de Bell et LaPadula ou empêcher des conflits d'intêrets comme dans le modèle de la Muraille de Chine. D'ailleurs, une section de ce document est consacrée à la comparaison des modèles de Bell et LaPadula et de RBAC96, et pour ce faire, le modèle à base de treillis, caractéristique de celui de Bell et LaPadula, est exprimé dans le formalisme RBAC introduit. D'autre part, RBAC permet aussi une plus grande facilité d'administration, car l'attribution de permissions se fait directement sur les rôles, classes d'utilisateurs (notion stable), sans qu'il ne soit jamais besoin de considérer les utilisateurs ou les objets un à un.

Dans ce document, le modèle RBAC96 est formalisé dans le cadre sémantique introduit dans [**?**] et finalisé dans [**?**] qui permet la spécification et la définition de politiques de contrôle d'accès ainsi que la formalisation des notions de requêtes, de modèles de contrôle d'accès et d'implantations. Cette formalisation est présentée sur trois niveaux, chaque notion du cadre est instanciée pour le modèle RBAC96 et est illustrée par un exemple simple. Cette formalisation, présentée dans les sections 2, 3 et 4, correspond au premier objectif du stage. La section 6 illustre l'utilisation de cette formalisation sur un exemple concret.

Afin de valider la formalisation décrite ci-dessus nous la mécanisons

ensuite à l'aide de l'atelier Focal[**?**]. Focal fournit un environnement de développement muni de traits objets (héritage, redéfinition, ...) et permettant d'écrire avec le même langage, des spécifications, des programmes et des preuves. Cet atelier permet de passer progressivement, par raffinements successifs, de la spécification à l'implantation tout en garantissant que le programme obtenu est correct par rapport à sa spécification. L'atelier Focal a déjà été utilisé avec succès, entre autres, pour implanter une bibliothèque de calcul formel et pour spécifier et valider la politique de sécurité au sol des aéroports [**?**] et est actuellement utilisé pour implanter une bibliothèque de politiques de contrôle d'accès [**?**, **?**] (c'est d'ailleurs dans le cadre de ce projet que s'inscrit le stage). L'essentiel de l'implantation Focal mettant en œuvre la formalisation du modèle RBAC96 ainsi que son illustration sur un exemple concret est présentée dans la section 7. Cette implantation correspond au deuxième objectif du stage.

Comme nous l'avons dit, le modèle RBAC96 est réputé générique puisqu'il permet de « coder » d'autres modèles. Aussi, nous présentons un schéma de traduction des modèles LBAC (*Lattice Based Access Control*) à base de treillis de niveaux de sécurité vers les modèles RBAC à base de rôles. Dans [**?**], les grandes lignes d'un tel schéma sont esquissées. Comme nous le verrons, ce schéma de traduction nous permet d'établir formellement que le modèle de Bell et LaPadula est plus restrictif que le modèle RBAC96. Cette comparaison s'établit ici encore dans le cadre formel décrit dans [**?**]. Elle est présentée dans la section 5 et correspond au troisième et dernier objectif du stage.

2 Politiques de contrôle d'accès

Dans ce document, une politique de contrôle d'accès est définie sur des systèmes vus comme des machines abstraites à états. Dans cette section, nous introduisons les divers concepts mis en jeu dans la définition d'une politique de contrôle d'accès et nous montrons comment ces concepts génériques peuvent être instanciés pour définir la politique de contrôle d'accès à base de rôles RBAC96. Enfin, nous décrivons la mise en oeuvre de cette politique sur un exemple concret.

2.1 Entités

Sujets et Objets Le contrôle des accès d'un système permet de régir les accès entre deux types d'entités. On distingue en effet deux ensembles d'entités. Le premier est l'ensemble $\mathcal{S}$ des sujets, aussi appelés entités actives, qui initient les actions du système. Il peut s'agir d'utilisateurs, de processus, etc. Le second est l'ensemble $\mathcal{O}$ des objets, aussi appelés entités passives, sur lesquels les actions sont effectuées. Il peut s'agir de fichiers, de ressources, de processus, etc.

Sujets et objets de RBAC96 La politique RBAC96 permet de distinguer sujets et utilisateurs. On considère un ensemble d'utilisateurs, U, qui peuvent initier une ou plusieurs sessions dans le système considéré. C'est cette notion de session qui correspond à l'ensemble $\mathcal{S}$ des sujets. La fonction :

$$user : \mathcal{S} \to \mathsf{U}$$

permet d'associer un utilisateur à une session, c'est-à-dire à un sujet. Ainsi, un utilisateur n'est pas modélisé par un sujet et appartient à un paramètre (de sécurité), comme nous le verrons plus loin. Si la notion de sujet dépend de celle d'utilisateur, en revanche, l'ensemble $\mathcal{O}$ des objets considérés par RBAC96 est complètement arbitraire.

Exemple On introduit l'ensemble des sujets $\mathcal{S} = \{s_1, s_2, s_3, s_4\}$ et l'ensemble des objets $\mathcal{O} = \{o_1, o_2\}$.

2.2 Accès

Notion d'accès Nous caractérisons ici la notion d'accès. Pour cela, nous introduisons un ensemble $\mathcal{A}$ des modes d'accès qui permettent de spécifier la manière avec laquelle un sujet accède à un objet. Les modes d'accès classiques sont lire, écrire, exécuter, etc. Une approche possible consiste à décrire un accès par un triplet (s, o, x) exprimant qu'un sujet s accède à un objet o selon le mode x. On note $\mathbb{A}$ l'ensemble des accès :

$$\mathbb{A} = \mathcal{S} \times \mathcal{O} \times \mathcal{A}$$

Accès de RBAC96 RBAC96 introduit une notion de permission. Une permission met en relation des opérations, que nous pouvons identifier aux modes d'accès, et des objets. On note :

$$\mathsf{P} \subseteq \mathcal{A} \times \mathcal{O}$$

l'ensemble des permissions. Cet ensemble est seulement inclus dans le produit cartésien $\mathcal{A} \times \mathcal{O}$ car RBAC96 permet de considérer des modes d'accès spécifiques à certains objets. Par exemple, dans un contexte hospitalier, le mode d'accès $Creer\_DossierMedecin$ ne peut pas être associé à l'objet $Ordonnance$.

Exemple (suite) On introduit l'ensemble des permissions :

$$\mathsf{P} = \{(a_1, o_1), (a_1, o_2), (a_2, o_1), (a_2, o_2)\}$$

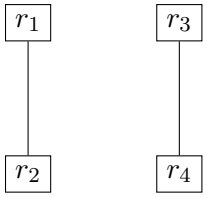
2.3 Paramètre de sécurité

Paramètre de sécurité d'une politique Les politiques de contrôle d'accès font intervenir des informations de sécurité associées aux sujets et/ou aux objets. Ces informations serviront à déterminer si un accès respecte ou non la politique. Ces informations peuvent être, par exemple, un ensemble d'habilitations ou de classes de conflits d'intérêt. Elles constituent ce que l'on appellera dans la suite le paramètre de sécurité de la politique. Il n'est bien sûr pas possible de décrire le « type » de ce paramètre à ce niveau de spécification. Nous noterons ce paramètre ρ.

Paramètre de sécurité de RBAC96 Le paramètre de sécurité ρ_{rbac} de RBAC96 regroupe un certain nombre de concepts. Tout d'abord, l'ensemble des utilisateurs U, vu précédemment, va y être inclus. Il fait partie du paramètre de sécurité dans la mesure où la notion d'utilisateur va servir à spécifier les rôles associés à un sujet. d'autre part, ρ_{rbac} spécifie la notion centrale d'un modèle RBAC, l'ensemble des rôles, noté R. Cet ensemble est muni d'une relation d'ordre partiel $\leq_{\mathsf{R}}$ (appelée aussi $\mathsf{RH} \subseteq \mathsf{R} \times \mathsf{R}$, *Role Hierarchy*). Finalement, le paramètre de sécurité ρ_{rbac} de RBAC96 s'écrit :

$$\rho_{\mathsf{rbac}} = (\mathsf{U}, \mathsf{R}, \leq_{\mathsf{R}})$$

Exemple (suite) On introduit l'ensemble des utilisateurs $\mathsf{U} = \{u_1, u_2, u_3\}$, l'ensemble des rôles $\mathsf{R} = \{r_1, r_2, r_3, r_4\}$ muni de l'ordre partiel $\leq_{\mathsf{R}}$ représenté sur la figure ci-dessous,



2.4 Etats

Les deux composantes d'un état Les systèmes sur lesquels s'appliquent les politiques de contrôle d'accès sont considérés, ici, comme des machines abstraites à états et l'on note Σ l'ensemble des états d'un système. Un état représente le système à un instant donné. Il décrit l'ensemble des accès courants du système. La fonction $\Lambda : \Sigma \to \wp(\mathbb{A})$ permet d'associer à chaque état σ appartenant à Σ, l'ensemble $\Lambda(\sigma)$ des accès courants du système dans cet état. De plus, un état contient des *fonctions de sécurité*, permettant de donner des informations concernant les paramètres de sécurité associées aux sujets et aux objets. Si l'on note SF l'ensemble des fonctions de sécurité, la fonction $\Upsilon : \Sigma \to \mathsf{SF}$ permet d'obtenir les fonctions de sécurité d'un état.

Fonctions de sécurité de RBAC96 L'ensemble des états de RBAC96 est noté Σ_{rbac}. Un état appartenant à Σ_{rbac} contient un ensemble d'accès courants m, et les fonctions de sécurité suivantes. Tout d'abord, la fonction *user*, vue précédemment, qui permet d'associer un utilisateur à un sujet (c'est-à-dire à une session). D'autre part, un rôle permet de faire le lien entre un utilisateur et les permissions qui lui sont associées. Cela est réalisé par les deux relations :

$$\begin{array}{ll} \mathsf{UA} \subseteq \mathsf{U} \times \mathsf{R} & (\textit{User Assignment}) \\ \mathsf{PA} \subseteq \mathsf{P} \times \mathsf{R} & (\textit{Permission Assignment}) \end{array}$$

associant respectivement un ou plusieurs rôles à un ou plusieurs utilisateurs et une ou plusieurs permissions à un ou plusieurs rôles. Enfin, la fonction :

$$roles : \mathcal{S} \rightarrow \wp(\mathsf{R})$$

définit l'ensemble des rôles actifs d'un sujet à un instant donné. Bien sûr, comme nous le verrons, l'ensemble des rôles actifs associés à un sujet est contraint par la relation UA et la fonction *user*.

Ainsi, un état σ appartenant à Σ_{rbac} s'écrit comme un 5-uplet :

$$\sigma = (\underbrace{m}_{\Lambda(\sigma)}, \underbrace{user, \mathsf{UA}, \mathsf{PA}, roles}_{\Upsilon(\sigma)})$$

Exemple (suite) Les fonctions de sécurité sont définies par :

$$\begin{array}{l} user(s_1) = user(s_2) = u_1 \quad user(s_3) = u_2 \quad user(s_4) = u_3 \\ \mathsf{UA} = \{(u_1, r_1), (u_2, r_2), (u_3, r_3)\} \\ \mathsf{PA} = \{((a_1, o_1), r_1), ((a_1, o_2), r_2), ((a_2, o_2), r_3), ((a_2, o_1), r_4)\} \\ roles(s_1) = roles(s_2) = \{r_1\} \quad roles(s_3) = \emptyset \quad roles(s_4) = \{r_4\} \end{array}$$

On remarque dans cet exemple que l'utilisateur u_1 est associé à deux sujets distincts s_1 et s_2.

2.5 Prédicat de sécurité

Etats sûrs Les états sûrs d'un système sont caractérisés par un prédicat de sécurité noté Ω. C'est ce prédicat qui définit la politique de contrôle d'accès. On utilise ici la logique du premier ordre pour définir ce prédicat. L'*ensemble des états sûrs* $\{\sigma \in \Sigma \mid \Omega(\sigma)\}$ est noté $\Sigma_{|\Omega}$.

Prédicat RBAC96 Le prédicat Ω_{rbac} caractérisant la *politique de sécurité* du modèle RBAC96 est défini comme suit. Tout d'abord, il impose que les rôles actifs associés à un sujet, c'est-à-dire à une session, appartiennent à

l'ensemble des rôles « autorisés » pour l'utilisateur qui a lancé cette session. Plus formellement :

$$\forall s \in \mathcal{S} \quad roles(s) \subseteq \mathsf{ER}(s, \mathsf{UA}) \tag{1}$$
$$\text{avec} \quad \mathsf{ER}(s, \mathsf{UA}) = \{r \in \mathsf{R} \mid \exists r' \in \mathsf{R} \quad r \leq_{\mathsf{R}} r' \wedge (user(s), r') \in \mathsf{UA}\}$$

Il est important de noter que la propriété (1) indique qu'un sujet peut activer un rôle qui n'est pas en relation avec son utilisateur suivant UA, grâce à l'ordre partiel $\leq_{\mathsf{R}}$. La propriété (1) introduit une contrainte sur les rôles actifs associés aux sujets et spécifiés par la fonction $roles$. Lors de la définition de cette fonction, plusieurs choix sont possibles. Par exemple, la définition suivante :

$$\forall s \in \mathcal{S} \quad roles_1(s) = \{r \in \mathsf{R} \mid (user(s), r) \in \mathsf{UA}\}$$

est une première possibilité. Elle décrit un ensemble qui contient tous les rôles les plus élevés dans la hiérarchie possibles pour un utilisateur donné. Une deuxième définition possible est :

$$\forall s \in \mathcal{S} \quad roles_2(s) = \{r \in \mathsf{R} \mid r \in \mathsf{ER}(s, \mathsf{UA})\}$$

On voit bien que $roles_1(s) \subseteq roles_2(s)$ étant donné que la fonction $roles_2(s)$ caractérise un ensemble qui contient tous les rôles possibles d'un utilisateur, aussi bien ceux en relation dans UA (comme pour $roles_1(s)$) que ceux qui leurs sont inférieurs selon la relation $\leq_{\mathsf{R}}$. Enfin, une troisième possibilité est de choisir une fonction $roles$ qui respecte le principe du moindre privilège en initialisant pour chaque sujet l'ensemble décrit par la fonction $roles$ à vide et en lui ajoutant un rôle soit en fonction de ses accès, soit à l'aide d'une requête explicite qui fixe ses besoins comme nous le verrons par la suite.

On définit ensuite la fonction :

$$\mathsf{EP} : \mathcal{S} \to \wp(\mathsf{P} \times \mathsf{R}) \to (\mathcal{S} \to \wp(\mathsf{R})) \to \wp(\mathsf{P})$$

qui fait correspondre à une session, une relation entre des permissions et des rôles et une fonction associant un ensemble de rôles à un sujet, toutes les permissions associées aux rôles actifs de cette session ainsi qu'aux « sous-rôles » des ces rôles actifs.

$$\forall s \in \mathcal{S} \;\; \mathsf{EP}(s, \mathsf{PA}, roles) = \bigcup_{r \in roles(s)} \{p \in \mathsf{P} \mid \exists r'' \in \mathsf{R} \; r'' \leq_{\mathsf{R}} r \wedge (p, r'') \in \mathsf{PA}\}$$

L'ensemble dénoté par $\mathsf{EP}(s, \mathsf{PA}, roles)$ définit pour un sujet s toutes les permissions associées à ses rôles actifs et aux rôles qui leurs sont inférieurs suivant $\leq_{\mathsf{R}}$. Pour qu'un état $\sigma = (m, user, \mathsf{UA}, \mathsf{PA}, roles)$ soit sûr, pour qu'il appartienne à $\Sigma_{\mathsf{rbac}_{|\Omega_{\mathsf{rbac}}}}$, il faut alors que chaque accès du système dans cet état soit permis :

$$\forall s \in \mathcal{S} \quad \forall o \in \mathcal{O} \quad \forall x \in \mathcal{A} \quad (s, o, x) \in m \Rightarrow (x, o) \in \mathsf{EP}(s, \mathsf{PA}, roles) \tag{2}$$

Ainsi, étant donné un état σ appartenant à Σ_{rbac}, $\Omega_{\mathsf{rbac}}(\sigma)$ est vérifié si les propriétés (1) et (2) sont satisfaites.

Exemple (suite)

$$\mathsf{ER}(s_1, \mathsf{UA}) = \mathsf{ER}(s_2, \mathsf{UA}) = \{r_1, r_2\}$$
$$\mathsf{ER}(s_3, \mathsf{UA}) = \{r_2\} \quad \mathsf{ER}(s_4, \mathsf{UA}) = \{r_3, r_4\}$$
$$\mathsf{EP}(s_1, \mathsf{PA}, roles) = \mathsf{EP}(s_2, \mathsf{PA}, roles) = \{(a_1, o_1), (a_1, o_2)\}$$
$$\mathsf{EP}(s_3, \mathsf{PA}, roles) = \emptyset \quad \mathsf{EP}(s_4, \mathsf{PA}, roles) = \{(a_2, o_1)\}$$

2.6 Politiques

En résumé, une politique de contrôle d'accès est définie de la manière suivante.

Définition 1 (Politique de contrôle d'accès) *Une politique de contrôle d'accès $\mathbb{P}[\rho]$, mettant en œuvre un paramètre de sécurité ρ, est définie par un 5-uplet $\mathbb{P}[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma, \Omega)$ où $\mathcal{S}$ est un ensemble de sujets, $\mathcal{O}$ est un ensemble d'objets, $\mathcal{A}$ est un ensemble de modes d'accès, Σ est l'ensemble des états du système sur lequel la politique est appliquée et Ω est le prédicat qui définit les états sûrs du système.*

Pour beaucoup de politiques de contrôle d'accès, la suppression d'un ou plusieurs accès d'un état sûr conduit encore à un état sûr. De telles politiques sont dites compactes et nous considérerons dans la suite de ce document uniquement des politiques compactes.

Définition 2 (Politique de contrôle d'accès compacte) *Une politique de contrôle d'accès $\mathbb{P}[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma, \Omega)$ est compacte si et seulement si :*

$$\forall \sigma_1 \in \Sigma \quad \Omega(\sigma_1) \Rightarrow (\forall \sigma_2 \in \Sigma \ (\Lambda(\sigma_2) \subseteq \Lambda(\sigma_1) \wedge \Upsilon(\sigma_1) = \Upsilon(\sigma_2)) \Rightarrow \Omega(\sigma_2))$$

De plus, on introduit la fonction $\mathcal{W} : \Sigma \rightarrow \wp(\wp(\mathbb{A}))$ qui permet de caractériser l'ensemble des accès qui peuvent être ajoutés de manière « sûre » dans un état sans changer les fonctions de sécurité.

Définition 3 (Accès potentiels d'un état)

$$\mathcal{W}(\sigma) = \left\{ \begin{array}{l} A \subseteq \wp(\mathbb{A}) \mid \forall \sigma' \in \Sigma \\ (\Upsilon(\sigma') = \Upsilon(\sigma) \wedge \Lambda(\sigma') = \Lambda(\sigma) \cup A) \Rightarrow \Omega(\sigma') \end{array} \right\}$$

A partir de ces définitions, on montre les trois lemmes suivants.

Lemme 1 $\mathcal{W}(\sigma) = \emptyset \Rightarrow \neg\Omega(\sigma)$

Preuve. Si $\mathcal{W}(\sigma) = \emptyset$ alors $\emptyset \notin \mathcal{W}(\sigma)$, et par définition de $\mathcal{W}$, il vient $\neg\Omega(\sigma)$. ◀

Lemme 2 *$\mathbb{P}[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma, \Omega)$ est compacte si et seulement si $\neg\Omega(\sigma) \Rightarrow \mathcal{W}(\sigma) = \emptyset$.*

PREUVE. ($\Rightarrow$). Supposons que la politique $\mathbb{P}[\rho]$ est compacte et considérons un état σ_1 tel que $\mathcal{W}(\sigma_1) \neq \emptyset$. Par définition de $\mathcal{W}$, il existe un ensemble d'accès A tel que tout état σ_2 satisfaisant $\Upsilon(\sigma_1) = \Upsilon(\sigma_2)$ et $\Lambda(\sigma_2) = \Lambda(\sigma_1) \cup A$ est sûr. Ainsi, comme $\Omega(\sigma_2)$ est vérifié et $\Lambda(\sigma_1) \subseteq \Lambda(\sigma_2)$, on en déduit que $\Omega(\sigma_1)$ est satisfait étant donné que $\mathbb{P}[\rho]$ est compacte.
($\Leftarrow$). Considérons un état sûr σ_1 et un état σ_2 tel que $\Lambda(\sigma_2) \subseteq \Lambda(\sigma_1)$ et $\Upsilon(\sigma_1) = \Upsilon(\sigma_2)$. Montrons que $\Omega(\sigma_2)$ est satisfait. Supposons $\neg\Omega(\sigma_2)$ et donc, par hypothèse, $\mathcal{W}(\sigma_2) = \emptyset$. Or, par définition de $\mathcal{W}$, $\Lambda(\sigma_1) \backslash \Lambda(\sigma_2)$ appartient à $\mathcal{W}(\sigma_2)$, étant donné que $\Omega(\sigma_1)$ est satisfait, ce qui mène à une contradiction. ◀

Lemme 3 *Soit $\mathbb{P}[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma, \Omega)$ une politique de contrôle d'accès.*

$$\forall \sigma_1, \sigma_2, \sigma_1', \sigma_2' \in \Sigma \quad \forall A \in \mathbb{A}$$
$$\left(\begin{array}{ll} & \mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2) \\ \wedge & \Lambda(\sigma_1') = \Lambda(\sigma_1) \cup A \wedge \Upsilon(\sigma_1') = \Upsilon(\sigma_1) \\ \wedge & \Lambda(\sigma_2') = \Lambda(\sigma_2) \cup A \wedge \Upsilon(\sigma_2') = \Upsilon(\sigma_2) \end{array} \right) \Rightarrow \mathcal{W}(\sigma_1') = \mathcal{W}(\sigma_2')$$

PREUVE. Considérons deux états σ_1 et σ_2 tels que $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$. De plus, considérons deux états σ_1' et σ_2', et un ensemble d'accès $A \subseteq \mathbb{A}$ tels que $\Lambda(\sigma_1') = \Lambda(\sigma_1) \cup A$, $\Upsilon(\sigma_1') = \Upsilon(\sigma_1)$, $\Lambda(\sigma_2') = \Lambda(\sigma_2) \cup A$ et $\Upsilon(\sigma_2') = \Upsilon(\sigma_2)$. Montrons que $\mathcal{W}(\sigma_1') \subseteq \mathcal{W}(\sigma_2')$ et que $\mathcal{W}(\sigma_2') \subseteq \mathcal{W}(\sigma_1')$.
1. $\mathcal{W}(\sigma_1') \subseteq \mathcal{W}(\sigma_2')$.
Considérons un ensemble d'accès A' appartenant $\mathcal{W}(\sigma_1')$ et un état σ_1'' tels que $\Lambda(\sigma_1'') = \Lambda(\sigma_1') \cup A'$, $\Upsilon(\sigma_1'') = \Upsilon(\sigma_1')$. Par définition de $\mathcal{W}$, $\Omega(\sigma_1'')$ est satisfait. Or $\Lambda(\sigma_1'') = \Lambda(\sigma_1) \cup A \cup A'$ et $\Upsilon(\sigma_1'') = \Upsilon(\sigma_1)$. On en déduit que $A \cup A'$ appartient à $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$. De plus, considérons un état σ_2'' tel que $\Lambda(\sigma_2'') = \Lambda(\sigma_2) \cup A \cup A'$, $\Upsilon(\sigma_2'') = \Upsilon(\sigma_2)$. On en déduit que $\Omega(\sigma_2'')$ est vérifié. Puisque $\Lambda(\sigma_2'') = \Lambda(\sigma_2') \cup A'$ et $\Upsilon(\sigma_2'') = \Upsilon(\sigma_2')$, on en conclut que A' appartient à $\mathcal{W}(\sigma_2')$, et donc que $\mathcal{W}(\sigma_1') \subseteq \mathcal{W}(\sigma_2')$.
2. $\mathcal{W}(\sigma_2') \subseteq \mathcal{W}(\sigma_1')$.
Un raisonnement similaire au point 1. permet de conclure. ◀

On introduit également la fonction $\mathcal{W}_\emptyset : \Sigma \rightarrow \wp(\wp(\mathbb{A}))$, qui étant donné un état, retourne tous les ensembles d'accès « compatibles » avec les fonctions de sécurité de cet état.

Définition 4 *$\mathcal{W}_\emptyset(\sigma) = \mathcal{W}(\sigma')$ avec $\Lambda(\sigma') = \emptyset$ et $\Upsilon(\sigma') = \Upsilon(\sigma)$.*

Il est possible de déduire de cette définition et du lemme précédent le lemme suivant.

Lemme 4 *Soit $\mathbb{P}[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma, \Omega)$ une politique de contrôle d'accès.*

$$\forall \sigma_1, \sigma_2 \in \Sigma \quad (\mathcal{W}_\emptyset(\sigma_1) = \mathcal{W}_\emptyset(\sigma_2) \wedge \Lambda(\sigma_1) = \Lambda(\sigma_2)) \Rightarrow \mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$$

PREUVE. Considérons deux états σ_1 et σ_2 tels que $\mathcal{W}_\emptyset(\sigma_1) = \mathcal{W}_\emptyset(\sigma_2)$ et $\Lambda(\sigma_1) = \Lambda(\sigma_2)$. Montrons que $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$. Considérons deux états σ_1' et σ_2' tels que $\Lambda(\sigma_1') = \emptyset$, $\Upsilon(\sigma_1') = \Upsilon(\sigma_1)$, $\Lambda(\sigma_2') = \emptyset$ et $\Upsilon(\sigma_2') = \Upsilon(\sigma_2)$. Par définition de $\mathcal{W}_\emptyset$, $\mathcal{W}(\sigma_1') = \mathcal{W}(\sigma_2')$. D'après le lemme 3, en ajoutant n'importe quel ensemble d'accès à σ_1' et σ_2', on obtient deux états σ_1'' et σ_2'' tels que $\mathcal{W}(\sigma_1'') = \mathcal{W}(\sigma_2'')$. Ainsi, il suffit d'ajouter l'ensemble d'accès $\Lambda(\sigma_1) = \Lambda(\sigma_2)$ pour conclure que $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$. ◀

Politique RBAC96 On définit formellement la politique de contrôle d'accès RBAC96 comme suit :

$$\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma_{\mathsf{rbac}}, \Omega_{\mathsf{rbac}})$$

On montre tout d'abord que cette politique RBAC96 est compacte.

Proposition 1 $\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}]$ *est compacte.*

PREUVE. Considérons deux états de Σ_{rbac}, $\sigma_1 = (m_1, user, \mathsf{UA}, \mathsf{PA}, roles)$ et $\sigma_2 = (m_2, user, \mathsf{UA}, \mathsf{PA}, roles)$ tels que $\Omega_{\mathsf{rbac}}(\sigma_1)$ soit satisfait et $m_2 \subseteq m_1$. Etant donné que $\Omega_{\mathsf{rbac}}(\sigma_1)$ est vérifié, pour tout sujet s, $roles(s) \subseteq \mathsf{ER}(s, \mathsf{UA})$. De plus, comme tout accès (s, o, x) appartenant à m_2 appartient à m_1, on en déduit que (x, o) appartient à $\mathsf{EP}(s, \mathsf{PA}, roles)$. Donc, $\Omega_{\mathsf{rbac}}(\sigma_2)$ est satisfait. ◀

Nous allons maintenant montrer que deux états sûrs ayant les mêmes fonctions de sécurité ont les mêmes accès potentiels.

Lemme 5 $\forall \sigma_1, \sigma_2 \in \Sigma_{\mathsf{rbac}}$

$$(\Omega_{\mathsf{rbac}}(\sigma_1) \wedge \Omega_{\mathsf{rbac}}(\sigma_2) \wedge \Upsilon(\sigma_1) = \Upsilon(\sigma_2)) \Rightarrow \mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$$

PREUVE. Considérons deux états de Σ_{rbac}, $\sigma_1 = (m_1, user, \mathsf{UA}, \mathsf{PA}, roles)$ et $\sigma_2 = (m_2, user, \mathsf{UA}, \mathsf{PA}, roles)$ tels que $\Omega_{\mathsf{rbac}}(\sigma_1)$ et $\Omega_{\mathsf{rbac}}(\sigma_2)$ soient satisfaits. Afin de démontrer que $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$, nous allons montrer l'inclusion dans les deux sens.
1. $\mathcal{W}(\sigma_1) \subseteq \mathcal{W}(\sigma_2)$.
Considérons E un ensemble d'accès tel que E appartient à $\mathcal{W}(\sigma_1)$. Posons $\sigma_1' = (m_1 \cup E, user, \mathsf{UA}, \mathsf{PA}, roles)$ et $\sigma_2' = (m_2 \cup E, user, \mathsf{UA}, \mathsf{PA}, roles)$. Par définition de $\mathcal{W}$, $\Omega_{\mathsf{rbac}}(\sigma_1')$ est vérifié. Nous voulons montrer que E appartient à $\mathcal{W}(\sigma_2)$, c'est-à-dire que $\Omega_{\mathsf{rbac}}(\sigma_2')$ est satisfait. Etant donné que $\Omega_{\mathsf{rbac}}(\sigma_1')$ est vérifié, pour tout sujet s, $roles(s) \subseteq \mathsf{ER}(s, \mathsf{UA})$ et donc par hypothèse σ_2' vérifie la propriété (1). D'autre part, étant donné que $\Omega_{\mathsf{rbac}}(\sigma_2)$ est satisfait, pour tout accès (s, o, x) appartenant à m_2, (x, o) appartient à $\mathsf{EP}(s, \mathsf{PA}, roles)$ et comme $\Omega_{\mathsf{rbac}}(\sigma_1')$ est satisfait, pour tout accès (s, o, x) appartenant à $m_1 \cup E$, (x, o) appartient à $\mathsf{EP}(s, \mathsf{PA}, roles)$ et donc pour tout accès (s, o, x) appartenant à E, (x, o) appartient à $\mathsf{EP}(s, \mathsf{PA}, roles)$.

Ainsi, pour tout accès (s, o, x) appartenant à $m_2 \cup E$, (x, o) appartient à $\mathsf{EP}(s, \mathsf{PA}, roles)$, ce qui permet de déduire que σ_2' vérifie la propriété (2) et donc E appartient à $\mathcal{W}(\sigma_2)$.
2. $\mathcal{W}(\sigma_2) \subseteq \mathcal{W}(\sigma_1)$.
Un raisonnement similaire au point 1. permet de conclure. ◀

3 Requêtes

3.1 Syntaxe

La notion de politique de contrôle d'accès introduite à la section précédente permet de spécifier quelles sont les entités du système et quels sont les états sûrs du système. Les transformations des états du système sont dûes à un ou plusieurs sujets qui soumettent une ou plusieurs requêtes afin d'accèder aux objets. On note $\mathcal{R}$ l'ensemble des requêtes.

Requêtes d'accès En général, on considère deux types de requêtes :

- le sujet s demande d'exercer une action selon la permission $p = (x, o)$, c'est-à-dire selon le mode d'accès x sur l'objet o, notée $\langle +, s, o, x\rangle$;
- le sujet s demande d'arrêter une action selon la permission $p = (x, o)$, c'est-à-dire selon le mode d'accès x sur l'objet o, notée $\langle -, s, o, x\rangle$.

On note $\mathcal{R}^{acc}$ cet ensemble de requêtes :

$$\mathcal{R}^{acc} = \{\langle +, s, o, x\rangle, \langle -, s, o, x\rangle\} \tag{3}$$

Requêtes administratives Pour RBAC96, on considère, en plus des requêtes de l'ensemble $\mathcal{R}^{acc}$ défini en (3), des requêtes *administratives* :

- le sujet s ajoute le rôle r aux rôles possibles de l'utilisateur u, notée $\langle +_{\mathsf{UA}}, s, u, r\rangle$;
- le sujet s enlève le rôle r des rôles possibles de l'utilisateur u, notée $\langle -_{\mathsf{UA}}, s, u, r\rangle$;
- le sujet s ajoute la permission p au rôle r, notée $\langle +_{\mathsf{PA}}, s, p, r\rangle$;
- le sujet s enlève la permission p au rôle r, notée $\langle -_{\mathsf{PA}}, s, p, r\rangle$;
- le sujet s ajoute le rôle r aux rôles actifs du sujet s', notée $\langle +_{roles}, s, s', r\rangle$;
- le sujet s enlève le rôle r des rôles actifs du sujet s', notée $\langle -_{roles}, s, s', r\rangle$.

Ainsi, pour RBAC96, l'ensemble des requêtes $\mathcal{R}_{\mathsf{rbac}}$ est le suivant :

$$\begin{array}{rcl} \mathcal{R}_{\mathsf{rbac}} & = & \mathcal{R}^{acc} \cup \mathcal{R}^{adm} \\ \text{avec} \quad \mathcal{R}^{adm} & = & \left\{ \begin{array}{l} \langle +_{\mathsf{UA}}, s, u, r\rangle, \langle -_{\mathsf{UA}}, s, u, r\rangle, \\ \langle +_{\mathsf{PA}}, s, p, r\rangle, \langle -_{\mathsf{PA}}, s, p, r\rangle, \\ \langle +_{roles}, s, s', r\rangle, \langle -_{roles}, s, s', r\rangle \end{array} \right\} \end{array} \tag{4}$$

Exemple (suite) Un exemple de requête est $\langle +, s_1, o_1, a_1\rangle$.

3.2 Sémantique

Etant donné l'ensemble $\mathcal{R}$ de requêtes, la « signification » de ses éléments doit être spécifiée. En effet, raisonner sur les implantations d'une politique n'a de sens que si l'on dispose d'une sémantique pour les requêtes. Ainsi, la relation :

$$[\![\mathcal{R}]\!]_\Sigma \subseteq \mathcal{R} \times \Sigma$$

permet de spécifier la sémantique des requêtes. Etant donnés une requête R et un état σ, l'énoncé $(R, \sigma) \in [\![\mathcal{R}]\!]_\Sigma$ permet de caractériser les propriétés qu'un état σ doit satisfaire quand il est obtenu à partir d'un état en appliquant avec succès la requête R. Par exemple, la sémantique de l'ensemble $\mathcal{R}^{acc}$ (défini en (3)) peut être exprimée de la manière suivante :

$$\begin{array}{l} (\langle +, s, o, x\rangle, \sigma) \in [\![\mathcal{R}^{acc}]\!]_\Sigma \Leftrightarrow (s, o, x) \in \Lambda(\sigma) \\ (\langle -, s, o, x\rangle, \sigma) \in [\![\mathcal{R}^{acc}]\!]_\Sigma \Leftrightarrow (s, o, x) \notin \Lambda(\sigma) \end{array} \tag{5}$$

Parallèlement la partition suivante de $\mathcal{R}$ est introduite :

$$\mathcal{R} = \mathcal{R}^{\oplus} \cup \mathcal{R}^{\ominus} \cup \mathcal{R}^{\oplus} \tag{6}$$

Cette partition permet de spécifier la variation des accès potentiels lors de l'application des requêtes. L'ensemble $\mathcal{R}^{\oplus}$ (resp. $\mathcal{R}^{\ominus}$) contient les requêtes élargissant (resp. rétrécissant), au sens de l'inclusion, les accès potentiels tandis que $\mathcal{R}^{\oplus}$ contient les autres requêtes. Par exemple, si le fait d'appliquer avec succès une requête R appartenant à $\mathcal{R}^{\ominus}$ à partir d'un état σ_1 conduit à un état σ_2 alors on doit avoir $\mathcal{W}(\sigma_2) \subseteq \mathcal{W}(\sigma_1)$.

Sémantique des requêtes administratives de RBAC96 On reprend la sémantique des requêtes (définie en (5)) en la complétant pour qu'elle prenne en compte l'ensemble des requêtes $\mathcal{R}_{\mathsf{rbac}} = \mathcal{R}^{acc} \cup \mathcal{R}^{adm}$ (défini en (4)).

La relation $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}} \subseteq \mathcal{R}_{\mathsf{rbac}} \times \Sigma_{\mathsf{rbac}}$ est définie comme suit.

$$
\begin{array}{lll}
(\langle +, s, o, x\rangle, \sigma) \in [\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}} & \Leftrightarrow & (s, o, x) \in \Lambda(\sigma) \\
(\langle -, s, o, x\rangle, \sigma) \in [\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}} & \Leftrightarrow & (s, o, x) \notin \Lambda(\sigma) \\
 & & \\
(\langle +_{\mathsf{UA}}, s, u, r\rangle, \sigma) \in [\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}} & \Leftrightarrow & (u, r) \in \mathsf{UA} \\
(\langle -_{\mathsf{UA}}, s, u, r\rangle, \sigma) \in [\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}} & \Leftrightarrow & (u, r) \notin \mathsf{UA} \\
 & & \\
(\langle +_{\mathsf{PA}}, s, p, r\rangle, \sigma) \in [\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}} & \Leftrightarrow & (p, r) \in \mathsf{PA} \\
(\langle -_{\mathsf{PA}}, s, p, r\rangle, \sigma) \in [\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}} & \Leftrightarrow & (p, r) \notin \mathsf{PA} \\
 & & \\
(\langle +_{roles}, s, s', r\rangle, \sigma) \in [\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}} & \Leftrightarrow & r \in roles(s') \\
(\langle -_{roles}, s, s', r\rangle, \sigma) \in [\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}} & \Leftrightarrow & r \notin roles(s')
\end{array}
\tag{7}
$$

Nous définissons à présent la partition introduite en (6) pour l'ensemble $\mathcal{R}_{\mathsf{rbac}}$. D'après le lemme 5, les requêtes $\langle +, s, o, x\rangle$ et $\langle -, s, o, x\rangle$ peuvent appartenir indifféremment aux ensembles $\mathcal{R}^{\ominus}_{\mathsf{rbac}}$, $\mathcal{R}^{\oslash}_{\mathsf{rbac}}$ ou $\mathcal{R}^{\oplus}_{\mathsf{rbac}}$. En effet, RBAC96 permet de considérer chaque accès indépendemment des autres accès courants ce qui assure l'absence de tout élargissement ou rétrécissement des accès potentiels du système pour ces deux requêtes. Par exemple, dans un contexte hospitalier, un chirurgien écrivant un compte rendu opératoire peut dans le même temps consulter le dossier d'un patient ou effectuer toute autre action qui lui est possible sans aucune contrainte. Par contre, les requêtes $\langle +_{\mathsf{UA}}, s, u, r\rangle$, $\langle +_{\mathsf{PA}}, s, p, r\rangle$ et $\langle +_{roles}, s, s', r\rangle$ appartiennent à l'ensemble $\mathcal{R}^{\oslash}_{\mathsf{rbac}}$, car le fait d'ajouter un rôle ou une permission élargit bien le champ des accès potentiels. Inversemment, les requêtes $\langle -_{\mathsf{UA}}, s, u, r\rangle$, $\langle -_{\mathsf{PA}}, s, p, r\rangle$ et $\langle -_{roles}, s, s', r\rangle$ appartiennent à l'ensemble $\mathcal{R}^{\ominus}_{\mathsf{rbac}}$. Aussi, un partitionnement possible de $\mathcal{R}_{\mathsf{rbac}}$ est le suivant :

$$
\mathcal{R}_{\mathsf{rbac}} = \underbrace{\left\{ \begin{array}{c} \langle +, s, o, x\rangle, \\ \langle -_{\mathsf{UA}}, s, u, r\rangle, \\ \langle -_{\mathsf{PA}}, s, p, r\rangle, \\ \langle -_{roles}, s, s', r\rangle \end{array} \right\}}_{\mathcal{R}^{\ominus}_{\mathsf{rbac}}} \cup \underbrace{\left\{ \begin{array}{c} \langle -, s, o, x\rangle, \\ \langle +_{\mathsf{UA}}, s, u, r\rangle, \\ \langle +_{\mathsf{PA}}, s, p, r\rangle, \\ \langle +_{roles}, s, s', r\rangle \end{array} \right\}}_{\mathcal{R}^{\oslash}_{\mathsf{rbac}}} \cup \underbrace{\emptyset}_{\mathcal{R}^{\oplus}_{\mathsf{rbac}}}
\tag{8}
$$

En fait, pour beaucoup de politiques classiques de contrôle d'accès, l'ajout (resp. le retrait) d'un accès rétrécit (resp. élargit) le spectre des accès futurs autorisés, c'est-à-dire que pour ces politiques la requête $\langle +, s, o, x\rangle$ appartient à $\mathcal{R}^{\ominus}$ (resp. $\langle -, s, o, x\rangle$ appartient à $\mathcal{R}^{\oslash}$). C'est pourquoi, afin de pouvoir comparer ultérieurement ces politiques avec la politique RBAC96, nous conservons ce partitionnement pour l'ensemble $\mathcal{R}_{\mathsf{rbac}}$.

4 Modèles de contrôle d'accès et implantations

4.1 Modèles et Implantations

A partir des définitions d'une politique de contrôle d'accès et d'un ensemble de requêtes muni d'une sémantique, un modèle de contrôle d'accès est défini comme suit.

Définition 5 (Modèle de contrôle d'accès) *Etant donné un paramètre de sécurité ρ, un modèle de contrôle d'accès $\mathbb{M}[\rho]$ est défini par un couple $\mathbb{M}[\rho] = (\mathbb{P}[\rho], [\![\mathcal{R}]\!]_\Sigma)$ où $\mathbb{P}[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma, \Omega)$ est une politique de contrôle d'accès, $\mathcal{R}$ est un ensemble de requêtes, et $[\![\mathcal{R}]\!]_\Sigma \subseteq \mathcal{R} \times \Sigma$ est une relation spécifiant la sémantique des requêtes.*

Implanter un modèle de contrôle d'accès $\mathbb{M}[\rho]$ par une machine abstraite à états consiste à définir un ensemble Σ_I d'*états initiaux* et une *fonction de transition* τ qui permet de passer d'un état du système à un autre suivant une requête dans $\mathcal{R}$. On peut ainsi donner la définition d'une implantation.

Définition 6 (Implantation) *Une implantation d'un modèle de contrôle d'accès $\mathbb{M}[\rho] = (\mathbb{P}[\rho], [\![\mathcal{R}]\!]_\Sigma)$ basé sur une politique $\mathbb{P}[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma, \Omega)$ est un couple (τ, Σ_I) où Σ_I est un ensemble d'états initiaux et où $\tau : \mathcal{R} \times \Sigma \to \mathcal{D} \times \Sigma$ est une fonction de transition avec $\mathcal{D} = \{\mathsf{yes}, \mathsf{no}\}$.*

On note $\Gamma_\tau^n(I)$ (resp. $\Gamma_\tau(I)$), l'ensemble des états atteignables du système en appliquant n fois (resp. un nombre fini de fois) la fonction τ depuis un état initial σ appartenant à I. Ainsi, il est possible de définir des propriétés pouvant être satisfaites par une implantation ou une fonction de transition.

— l'implantation (τ, Σ_I) est dite $\mathbb{P}[\rho]$-*correcte* si et seulement si chaque état atteignable depuis un état initial est sûr :

$$\mathbb{P}[\rho] \vdash (\tau, \Sigma_I) \Leftrightarrow \Gamma_\tau(\Sigma_I) \subseteq \Sigma_{|\Omega}$$

— la fonction de transition τ est dite $[\![\mathcal{R}]\!]_\Sigma$-*correcte*, noté $[\![\mathcal{R}]\!]_\Sigma \vdash \tau$ si et seulement si :

$$\forall \sigma_1, \sigma_2 \in \Sigma \quad \forall R \in \mathcal{R} \quad \tau(R, \sigma_1) = (\mathsf{yes}, \sigma_2) \Rightarrow (R, \sigma_2) \in [\![\mathcal{R}]\!]_\Sigma$$

— la fonction de transition τ est dite $\mathcal{W}$-*conforme* si et seulement si :

$$\begin{array}{l} \forall \sigma_1, \sigma_2 \in \Sigma \quad \forall d \in \mathcal{D} \quad \forall R \in \mathcal{R} \\ \tau(R, \sigma_1) = (d, \sigma_2) \Rightarrow \\ \left(\begin{array}{rl} & d = \mathsf{yes} \Rightarrow \left(\begin{array}{rl} & R \in \mathcal{R}^{\ominus} \Rightarrow \mathcal{W}(\sigma_1) \subseteq \mathcal{W}(\sigma_2) \\ \wedge & R \in \mathcal{R}^{\ominus} \Rightarrow \mathcal{W}(\sigma_2) \subseteq \mathcal{W}(\sigma_1) \end{array} \right) \\ \wedge & d = \mathsf{no} \Rightarrow \mathcal{W}(\sigma_2) \subseteq \mathcal{W}(\sigma_1) \end{array} \right) \end{array}$$

— Etant donné un modèle de contrôle d'accès $\mathbb{M}[\rho] = (\mathbb{P}[\rho], [\![\mathcal{R}]\!]_\Sigma)$, l'implantation (τ, Σ_I) est dite $\mathbb{M}[\rho]$-*correcte* si et seulement si elle est $\mathbb{P}[\rho]$-*correcte* et que τ est $[\![\mathcal{R}]\!]_\Sigma$-*correcte* et $\mathcal{W}$-*conforme*. Dans ce cas, on écrit $\mathbb{M}[\rho] \vdash (\tau, \Sigma_I)$.

Le lemme suivant permet de montrer qu'une implantation est correcte par rapport à une politique de contrôle d'accès.

Lemme 6 *L'implantation (τ, Σ_I) d'un modèle basé sur une politique $\mathbb{P}[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma, \Omega)$ est $\mathbb{P}[\rho]$-correcte si :*

$$\begin{array}{ll} & \Sigma_I \subseteq \Sigma_{|\Omega} \\ \wedge & \forall \sigma, \sigma' \in \Sigma \quad \forall R \in \mathcal{R} \quad \forall d \in \mathcal{D} \quad (\Omega(\sigma) \wedge \tau(R, \sigma) = (d, \sigma')) \Rightarrow \Omega(\sigma') \end{array}$$

Preuve. Montrons par induction sur n que les états accessibles sont inclus dans les états sûrs :

$$\forall n \in \mathbb{N} \quad \forall \sigma'' \in \Gamma_\tau^n(\Sigma_I) \quad \Omega(\sigma'')$$

$*$ Si $n = 0$ alors σ'' appartient à $\Sigma_I \subseteq \Sigma_{|\Omega}$ et donc $\Omega(\sigma'')$ est satisfait.
$*$ Si $n = k + 1$ alors il existe un état σ_1 appartenant à $\Gamma_\tau^k(\Sigma_I)$ tel que pour une requête R_1 et une réponse d_1, $\tau(R_1, \sigma_1) = (d_1, \sigma'')$ et, par hypothèse d'induction, $\Omega(\sigma_1)$ est vérifié. Ainsi, par hypothèse, $\Omega(\sigma'')$ est satisfait, ce qui nous permet de conclure. ◀

Nous caractérisons maintenant les implantations qui « préservent » une relation d'équivalence sur les états (i.e. les fonctions de transition qui appliquées à deux états équivalents renvoient la même réponse et deux états encore équivalents).

Définition 7 *Une fonction de transition τ est dite $\equiv$-préservante, ce que nous notons $\equiv \vdash \tau$, si et seulement si :*

$$\forall \sigma_1, \sigma_2, \sigma_1', \sigma_2' \in \Sigma \quad \forall R \in \mathcal{R} \quad \forall d \in \mathcal{D}$$
$$\left(\begin{array}{ll} & \sigma_1 \equiv \sigma_2 \\ \wedge & \tau(R, \sigma_1) = (d_1, \sigma_1') \\ \wedge & \tau(R, \sigma_2) = (d_2, \sigma_2') \end{array} \right) \Rightarrow \left(\begin{array}{ll} & \sigma_1' \equiv \sigma_2' \\ \wedge & d_1 = d_2 \end{array} \right)$$

De plus, la notion de modèles réduits est introduite. Elle consiste essentiellement à considérer les classes d'équivalence d'états plutôt que les états. Etant donnés un état σ et une relation d'équivalence $\equiv$ sur les états, $[\sigma]$ représente la classe d'équivalence de σ et $\Sigma_{/\equiv}$ représente l'ensemble quotient par rapport à $\equiv$ de Σ. On note $e(\sigma)$ l'élément canonique associé à σ lorsque la relation d'équivalence est définie conjointement avec une fonction de projection $e : \Sigma \to \Sigma$ telle que :

$$\forall \sigma, \sigma' \in \Sigma \quad \sigma \equiv \sigma' \Leftrightarrow e(\sigma) = e(\sigma')$$

et, étant donné un ensemble E d'états, on note $\hat{e}(E) = \{e(\sigma) \mid \sigma \in E\}$.

Ainsi, nous pouvons donner les définitions de politiques et de modèles réduits.

Définition 8 *Etant donnés $\mathbb{P}[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma, \Omega)$ une politique de contrôle d'accès, $\mathbb{M}[\rho] = (\mathbb{P}[\rho], [\![\mathcal{R}]\!]_\Sigma)$ un modèle de contrôle d'accès et $\equiv$ une relation d'équivalence sur Σ définie conjointement avec une fonction de projection $e : \Sigma \to \Sigma$ telle que :*

$$(\sigma_1 \equiv \sigma_2 \wedge \Omega(\sigma_1)) \Rightarrow \Omega(\sigma_2)$$
$$\forall R \in \mathcal{R} \quad (\sigma_1 \equiv \sigma_2 \wedge (R, \sigma_1) \in [\![\mathcal{R}]\!]_\Sigma) \Rightarrow (R, \sigma_2) \in [\![\mathcal{R}]\!]_\Sigma$$

— *La réduction de la politique $\mathbb{P}[\rho]$ selon la relation d'équivalence $\equiv$ est la politique $\mathbb{P}^\sharp_\equiv[\rho] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \hat{e}(\Sigma), \Omega)$.*

— *La réduction du modèle $\mathbb{M}[\rho]$ selon la relation d'équivalence $\equiv$ est le modèle $\mathbb{M}^\sharp_\equiv[\rho] = (\mathbb{P}^\sharp_\equiv[\rho], [\![\mathcal{R}]\!]_{\hat{e}(\Sigma)})$ où $[\![\mathcal{R}]\!]_{\hat{e}(\Sigma)} = \{(R, \sigma) \in [\![\mathcal{R}]\!]_\Sigma \mid \sigma \in \hat{e}(\Sigma)\}$.*

Enfin, on donne la définition de la relation d'équivalence sur les états que nous considérerons par la suite.

Définition 9 *Etant donnés $\mathbb{M}[\rho] = (\mathbb{P}[\rho], [\![\mathcal{R}]\!]_\Sigma)$ un modèle de contrôle d'accès et deux états σ_1 et σ_2 appartenant à Σ.*

— $\sigma_1 \equiv_{\mathcal{W}} \sigma_2 \Leftrightarrow \mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$

— $\sigma_1 \equiv_{\mathcal{W}_\emptyset} \sigma_2 \Leftrightarrow \mathcal{W}_\emptyset(\sigma_1) = \mathcal{W}_\emptyset(\sigma_2)$

— $\sigma_1 \equiv_{\mathcal{R}} \sigma_2 \Leftrightarrow (\forall R \in \mathcal{R} \quad (\sigma_1, R) \in [\![\mathcal{R}]\!]_\Sigma \Leftrightarrow (\sigma_2, R) \in [\![\mathcal{R}]\!]_\Sigma)$

— $\equiv_\iota = (\equiv_{\mathcal{W}} \cap \equiv_{\mathcal{W}_\emptyset} \cap \equiv_{\mathcal{R}})$
($i.e.$ $\sigma_1 \equiv_\iota \sigma_2 \Leftrightarrow (\sigma_1 \equiv_{\mathcal{W}} \sigma_2 \wedge \sigma_1 \equiv_{\mathcal{W}_\emptyset} \sigma_2 \wedge \sigma_1 \equiv_{\mathcal{R}} \sigma_2)$)

Dans la suite, étant donnée une politique $\mathbb{P}[\rho]$ (resp. un modèle $\mathbb{M}[\rho]$), nous notons $\mathbb{P}^\sharp[\rho]$ la politique $\mathbb{P}^\sharp_{\equiv_\iota}[\rho]$ (resp. $\mathbb{M}^\sharp[\rho]$ le modèle $\mathbb{M}^\sharp_{\equiv_\iota}[\rho]$) afin d'alléger les notations.

4.2 Implantations d'un modèle à base de rôles

Toutes les définitions et les propriétés introduites dans la sous-section précédente permettent de définir formellement le modèle de contrôle d'accès de RBAC96 :

$$\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] = (\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}], [\![\mathcal{R}]\!]_{\Sigma_{\mathsf{rbac}}})$$

4.2.1 Fonction de transition sur les accès

Une première fonction de transition $\tau^{acc}_{\mathsf{rbac}}$, ne traitant que les requêtes appartenant à $\mathcal{R}^{acc}$, est définie dans la table 1.

La proposition suivante énonce le fait que l'implantation $(\tau^{acc}_{\mathsf{rbac}}, \Sigma^{\mathsf{rbac}}_I)$ est correcte.

$$\tau^{acc}_{\mathsf{rbac}}(R,(m,user,\mathsf{UA},\mathsf{PA},roles))$$

$$= \begin{cases} (\mathsf{yes},(m\cup\{(s,o,x)\},user,\mathsf{UA},\mathsf{PA},roles)) \\ \quad \text{si} \quad R=\langle +,s,o,x\rangle \\ \quad \wedge \quad (x,o)\in \mathsf{EP}(s,\mathsf{PA},roles) \\ \\ (\mathsf{yes},(m\setminus\{(s,o,x)\},user,\mathsf{UA},\mathsf{PA},roles)) \\ \quad \text{si} \quad R=\langle -,s,o,x\rangle \\ \\ (\mathsf{no},(m,user,\mathsf{UA},\mathsf{PA},roles)) \text{ sinon} \end{cases}$$

TABLE 1 – Fonction de transition $\tau^{acc}_{\mathsf{rbac}}$ pour RBAC96

Proposition 2 $\Sigma^{\mathsf{rbac}}_I \subseteq \Sigma_{\mathsf{rbac}_{|\Omega_{\mathsf{rbac}}}} \Rightarrow \mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \vdash (\tau^{acc}_{\mathsf{rbac}}, \Sigma^{\mathsf{rbac}}_I)$

PREUVE. Pour prouver que $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \vdash (\tau^{acc}_{\mathsf{rbac}}, \Sigma^{\mathsf{rbac}}_I)$, il faut montrer que la fonction $\tau^{acc}_{\mathsf{rbac}}$ est $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$-*correcte* et $\mathcal{W}$-*conforme* et que l'implantation $(\tau^{acc}_{\mathsf{rbac}}, \Sigma^{\mathsf{rbac}}_I)$ est $\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}]$-*correcte*.

• Montrons que $\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \vdash (\tau^{acc}_{\mathsf{rbac}}, \Sigma^{\mathsf{rbac}}_I)$.
Puisque $\Sigma^{\mathsf{rbac}}_I \subseteq \Sigma_{\mathsf{rbac}_{|\Omega_{\mathsf{rbac}}}}$, d'après le lemme 6, il suffit de montrer que :

$$\forall \sigma,\sigma' \in \Sigma_{\mathsf{rbac}} \quad \forall R \in \mathcal{R}^{acc} \quad \forall d \in \mathcal{D}$$
$$(\Omega_{\mathsf{rbac}}(\sigma) \wedge \tau^{acc}_{\mathsf{rbac}}(R,\sigma) = (d,\sigma')) \Rightarrow \Omega_{\mathsf{rbac}}(\sigma')$$

Posons $\sigma = (m, user, \mathsf{UA}, \mathsf{PA}, roles)$. Montrons que $\Omega_{\mathsf{rbac}}(\sigma')$ est satisfait, c'est-à-dire que σ' respecte les propriétés (1) et (2). Trois cas sont possibles :
1. $d = \mathsf{no}$.
Par définition de $\tau^{acc}_{\mathsf{rbac}}$, $\sigma = \sigma'$, ce qui nous permet de conclure que $\Omega_{\mathsf{rbac}}(\sigma')$ est vérifié par hypothèse.
2. $d = \mathsf{yes}$ et $R = \langle -,s,o,x\rangle$.
Par définition de $\tau^{acc}_{\mathsf{rbac}}$, $\sigma' = (m\setminus\{(s,o,x)\}, user, \mathsf{UA}, \mathsf{PA}, roles)$. Etant donné que $(m\setminus\{(s,o,x)\}) \subseteq m$, que la politique $\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}]$ est compacte d'après la proposition 1, et que par hypothèse $\Omega_{\mathsf{rbac}}(\sigma)$ est vérifié, on en déduit que $\Omega_{\mathsf{rbac}}(\sigma')$ est satisfait.
3. $d = \mathsf{yes}$ et $R = \langle +,s,o,x\rangle$.
Par définition de $\tau^{acc}_{\mathsf{rbac}}$, $\sigma' = (m\cup\{(s,o,x)\}, user, \mathsf{UA}, \mathsf{PA}, roles)$ et (x,o) appartient à $\mathsf{EP}(s,\mathsf{PA},roles)$. Par hypothèse, $\Omega_{\mathsf{rbac}}(\sigma)$ est satisfait, donc pour tout sujet s', $roles(s') \subseteq \mathsf{ER}(s',\mathsf{UA})$, et donc σ' vérifie la propriété (1). De plus, par hypothèse, pour tout accès (s'',o',x') appartenant à m, (x',o') appartient à $\mathsf{EP}(s'',\mathsf{PA},roles)$. Or, $\Lambda(\sigma') = (m\cup\{(s,o,x)\})$ et (x,o) appartient à $\mathsf{EP}(s,\mathsf{PA},roles)$, donc σ' vérifie la propriété (2). Ainsi, σ' vérifie les propriétés (1) et (2), donc $\Omega_{\mathsf{rbac}}(\sigma')$ est satisfait.

• Montrons que $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}} \vdash \tau^{acc}_{\mathsf{rbac}}$.
Considérons deux états dans Σ_{rbac}, $\sigma_1 = (m_1, user, \mathsf{UA}, \mathsf{PA}, roles)$ et $\sigma_2 = (m_2, user, \mathsf{UA}, \mathsf{PA}, roles)$, et une requête R appartenant à $\mathcal{R}^{acc}$ tels que $\tau^{acc}_{\mathsf{rbac}}(R, \sigma_1) = (\mathsf{yes}, \sigma_2)$. Une telle requête R existe car la fonction $\tau^{acc}_{\mathsf{rbac}}$ ne modifie pas les fonctions de sécurité d'un état. Montrons que (R, σ_2) appartient à $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$. Deux cas sont possibles, soit $R = \langle +, s, o, x\rangle$, soit $R = \langle -, s, o, x\rangle$.
Si $R = \langle +, s, o, x\rangle$ alors, d'après la définition de $\tau^{acc}_{\mathsf{rbac}}$, et étant donné que la réponse est yes, $\sigma_2 = (m_1 \cup \{(s, o, x)\}, user, \mathsf{UA}, \mathsf{PA}, roles)$, et donc (s, o, x) appartient à m_2, ce qui permet de conclure que $(\langle +, s, o, x\rangle, \sigma_2)$ appartient à $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$.
Si $R = \langle -, s, o, x\rangle$ alors $\sigma_2 = (m_1 \setminus \{(s, o, x)\}, user, \mathsf{UA}, \mathsf{PA}, roles)$, et donc (s, o, x) n'appartient pas à m_2, ce qui permet de conclure que $(\langle -, s, o, x\rangle, \sigma_2)$ appartient à $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$.

• Montrons que $\tau^{acc}_{\mathsf{rbac}}$ est $\mathcal{W}$-*conforme*.
Considérons deux états dans Σ_{rbac}, $\sigma_1 = (m_1, user, \mathsf{UA}, \mathsf{PA}, roles)$ et $\sigma_2 = (m_2, user, \mathsf{UA}, \mathsf{PA}, roles)$, une requête R appartenant à $\mathcal{R}^{acc}$ et une réponse d tels que $\tau^{acc}_{\mathsf{rbac}}(R, \sigma_1) = (d, \sigma_2)$. Une telle requête R existe car la fonction $\tau^{acc}_{\mathsf{rbac}}$ ne modifie pas les fonctions de sécurité d'un état. Quatre cas sont possibles :
1. $\Omega_{\mathsf{rbac}}(\sigma_1) \wedge \Omega_{\mathsf{rbac}}(\sigma_2)$.
Etant donné que $\Upsilon(\sigma_1) = \Upsilon(\sigma_2)$, d'après le lemme 5 il vient $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$ ce qui nous permet de conclure.
2. $\Omega_{\mathsf{rbac}}(\sigma_1) \wedge \neg\Omega_{\mathsf{rbac}}(\sigma_2)$.
Impossible car nous avons montré que $(\tau^{acc}_{\mathsf{rbac}}, \Sigma^{\mathsf{rbac}}_I)$ est $\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}]$-*correcte*.
3. $\neg\Omega_{\mathsf{rbac}}(\sigma_1) \wedge \Omega_{\mathsf{rbac}}(\sigma_2)$.
Dans ce cas, d'après les lemmes 1 et 2, $\mathcal{W}(\sigma_1) = \emptyset$ et $\mathcal{W}(\sigma_2) \neq \emptyset$. Ainsi, $\sigma_1 \neq \sigma_2$ et donc $d \neq \mathsf{no}$. De plus, si $R = \langle +, s, o, x\rangle$ alors $\Lambda(\sigma_2) = \Lambda(\sigma_1) \cup \{(s, o, x)\}$. Or, $\Omega_{\mathsf{rbac}}(\sigma_2)$ est vérifié et comme la politique $\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}]$ est compacte, on en déduit que $\Omega_{\mathsf{rbac}}(\sigma_1)$ est satisfait ce qui est en contradiction avec notre hypothèse. Ainsi, $d = \mathsf{yes}$, $R = \langle -, s, o, x\rangle$ et $\mathcal{W}(\sigma_1) \subseteq \mathcal{W}(\sigma_2)$ ce qui nous permet de conclure car $\langle -, s, o, x\rangle$ appartient à $\mathcal{R}^{\Subset}$.
4. $\neg\Omega_{\mathsf{rbac}}(\sigma_1) \wedge \neg\Omega_{\mathsf{rbac}}(\sigma_2)$.
Dans ce cas, d'après les lemmes 1 et 2, $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2) = \emptyset$ ce qui nous permet de conclure. ◀

4.2.2 Fonction de transition administrative

Une deuxième fonction de transition $\tau^{adm}_{\mathsf{rbac}}$, ne traitant que les requêtes appartenant à $\mathcal{R}^{adm}$, est définie dans la table 2. On peut remarquer que pour la requête $\langle -_{roles}, s, s', r\rangle$, on vérifie que le rôle r retiré aux rôles actifs du sujet s' appartient à ses rôles possibles ce qui peut paraître superflu mais cela nous permettra de montrer que la fonction $\tau^{adm}_{\mathsf{rbac}}$ est $\mathcal{W}$-*conforme*.

Montrons que l'implantation $(\tau^{adm}_{\mathsf{rbac}}, \Sigma^{\mathsf{rbac}}_I)$ est correcte.

$\tau^{adm}_{\mathsf{rbac}}(R, (m, user, \mathsf{UA}, \mathsf{PA}, roles))$

$$
= \begin{cases}
(\mathsf{yes}, (m, user, \mathsf{UA} \cup \{(u,r)\}, \mathsf{PA}, roles)) \\
\quad \text{si} \quad R = \langle +_{\mathsf{UA}}, s, u, r \rangle \\
\\
(\mathsf{yes}, (m, user, \mathsf{UA} \setminus \{(u,r)\}, \mathsf{PA}, roles)) \\
\quad \text{si} \quad R = \langle -_{\mathsf{UA}}, s, u, r \rangle \\
\quad \wedge \quad \forall s' \in \mathcal{S} \\
\qquad user(s') = u \Rightarrow \\
\qquad roles(s') \subseteq \mathsf{ER}(s', \mathsf{UA} \setminus \{(u,r)\}) \\
\\
(\mathsf{yes}, (m, user, \mathsf{UA}, \mathsf{PA} \cup \{(p,r)\}, roles)) \\
\quad \text{si} \quad R = \langle +_{\mathsf{PA}}, s, p, r \rangle \\
\\
(\mathsf{yes}, (m, user, \mathsf{UA}, \mathsf{PA} \setminus \{(p,r)\}, roles)) \\
\quad \text{si} \quad R = \langle -_{\mathsf{PA}}, s, p, r \rangle \\
\quad \wedge \quad \forall (s', o, x) \in m \\
\qquad p = (x, o) \Rightarrow \\
\qquad (x, o) \in \mathsf{EP}(s', \mathsf{PA} \setminus \{(p,r)\}, roles) \\
\\
(\mathsf{yes}, (m, user, \mathsf{UA}, \mathsf{PA}, roles[s' \leftarrow roles(s') \cup \{r\}])) \\
\quad \text{si} \quad R = \langle +_{roles}, s, s', r \rangle \\
\quad \wedge \quad r \in \mathsf{ER}(s', \mathsf{UA}) \\
\\
(\mathsf{yes}, (m, user, \mathsf{UA}, \mathsf{PA}, roles[s' \leftarrow roles(s') \setminus \{r\}])) \\
\quad \text{si} \quad R = \langle -_{roles}, s, s', r \rangle \\
\quad \wedge \quad r \in \mathsf{ER}(s', \mathsf{UA}) \\
\quad \wedge \quad \forall o \in \mathcal{O} \quad \forall x \in \mathcal{A} \\
\qquad (s', o, x) \in m \Rightarrow \\
\qquad (x, o) \in \mathsf{EP}(s', \mathsf{PA}, roles[s' \leftarrow roles(s') \setminus \{r\}]) \\
\\
(\mathsf{no}, (m, user, \mathsf{UA}, \mathsf{PA}, roles)) \text{ sinon}
\end{cases}
$$

TABLE 2 – Fonction de transition $\tau^{adm}_{\mathsf{rbac}}$ pour RBAC96

Proposition 3 $\Sigma_I^{\mathsf{rbac}} \subseteq \Sigma_{\mathsf{rbac}_{|\Omega_{\mathsf{rbac}}}} \Rightarrow \mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \vdash (\tau_{\mathsf{rbac}}^{adm}, \Sigma_I^{\mathsf{rbac}})$

PREUVE. Pour prouver que $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \vdash (\tau_{\mathsf{rbac}}^{adm}, \Sigma_I^{\mathsf{rbac}})$, il faut montrer que la fonction $\tau_{\mathsf{rbac}}^{adm}$ est $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$-*correcte* et $\mathcal{W}$-*conforme* et que l'implantation $(\tau_{\mathsf{rbac}}^{adm}, \Sigma_I^{\mathsf{rbac}})$ est $\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}]$-*correcte*.

• Montrons que $\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \vdash (\tau_{\mathsf{rbac}}^{adm}, \Sigma_I^{\mathsf{rbac}})$.
Puisque $\Sigma_I^{\mathsf{rbac}} \subseteq \Sigma_{\mathsf{rbac}_{|\Omega_{\mathsf{rbac}}}}$, d'après le lemme 6, il suffit de montrer que :

$$\forall \sigma, \sigma' \in \Sigma_{\mathsf{rbac}} \quad \forall R \in \mathcal{R}^{adm} \quad \forall d \in \mathcal{D}$$
$$(\Omega_{\mathsf{rbac}}(\sigma) \wedge \tau_{\mathsf{rbac}}^{adm}(R, \sigma) = (d, \sigma')) \Rightarrow \Omega_{\mathsf{rbac}}(\sigma')$$

Posons $\sigma = (m, user, \mathsf{UA}, \mathsf{PA}, roles)$. Montrons que $\Omega_{\mathsf{rbac}}(\sigma')$ est vrai, c'est-à-dire que σ' respecte les propriétés (1) et (2). Si $d = \mathsf{no}$ alors, par définition de $\tau_{\mathsf{rbac}}^{adm}$, $\sigma = \sigma'$, et on en conclut que $\Omega_{\mathsf{rbac}}(\sigma')$ est vérifié par hypothèse. Si $d = \mathsf{yes}$ alors six cas sont possibles :
1. $R = \langle +_{\mathsf{UA}}, s, u, r \rangle$.
Par définition de $\tau_{\mathsf{rbac}}^{adm}$, $\sigma' = (m, user, \mathsf{UA} \cup \{(u, r)\}, \mathsf{PA}, roles)$. Par hypothèse, $\Omega_{\mathsf{rbac}}(\sigma)$ est vérifié, donc pour tout sujet s', $roles(s') \subseteq \mathsf{ER}(s', \mathsf{UA})$. Or, pour tout sujet s', $\mathsf{ER}(s', \mathsf{UA}) \subseteq \mathsf{ER}(s', \mathsf{UA} \cup \{(u, r)\})$, donc σ' vérifie la propriété (1). De plus, par la même hypothèse, pour tout accès (s'', o, x) appartenant à m, (x, o) appartient à $\mathsf{EP}(s'', \mathsf{PA}, roles)$ ce qui nous permet de déduire que σ' vérifie la propriété (2). Ainsi, σ' vérifie les propriétés (1) et (2), donc $\Omega_{\mathsf{rbac}}(\sigma')$ est satisfait.
2. $R = \langle -_{\mathsf{UA}}, s, u, r \rangle$.
Par définition de $\tau_{\mathsf{rbac}}^{adm}$, $\sigma' = (m, user, \mathsf{UA} \setminus \{(u, r)\}, \mathsf{PA}, roles)$ et pour tout sujet s', si $user(s') = u$ alors $roles(s') \subseteq \mathsf{ER}(s', \mathsf{UA} \setminus \{(u, r)\})$. Par hypothèse, $\Omega_{\mathsf{rbac}}(\sigma)$ est vérifié, donc pour tout sujet s', $roles(s') \subseteq \mathsf{ER}(s', \mathsf{UA})$. Or, pour tout sujet s', si $user(s') = u$ alors $roles(s') \subseteq \mathsf{ER}(s', \mathsf{UA} \setminus \{(u, r)\})$, et donc pour tout sujet s', $roles(s') \subseteq \mathsf{ER}(s', \mathsf{UA}) \setminus \{(u, r)\})$, ce qui permet de déduire que σ' vérifie la propriété (1). De plus, par la même hypothèse, pour tout accès (s'', o, x) appartenant à m, (x, o) appartient à $\mathsf{EP}(s'', \mathsf{PA}, roles)$ ce qui nous permet de déduire que σ' vérifie la propriété (2). Ainsi, σ' vérifie les propriétés (1) et (2), donc $\Omega_{\mathsf{rbac}}(\sigma')$ est satisfait.
3. $R = \langle +_{\mathsf{PA}}, s, p, r \rangle$.
Par définition de $\tau_{\mathsf{rbac}}^{adm}$, $\sigma' = (m, user, \mathsf{UA}, \mathsf{PA} \cup \{(p, r)\}, roles)$. Par hypothèse, $\Omega_{\mathsf{rbac}}(\sigma)$ est vérifié, donc pour tout sujet s', $roles(s') \subseteq \mathsf{ER}(s', \mathsf{UA})$ ce qui nous permet de déduire que σ' vérifie la propriété (1). De plus, par la même hypothèse, pour tout accès (s'', o, x) appartenant à m, (x, o) appartient à $\mathsf{EP}(s'', \mathsf{PA}, roles)$. Or, pour tout sujet s', $\mathsf{EP}(s', \mathsf{PA}, roles) \subseteq \mathsf{EP}(s', \mathsf{PA} \cup \{(p, r)\}, roles)$, donc σ' vérifie la propriété (2). Ainsi, σ' vérifie les propriétés (1) et (2), donc $\Omega_{\mathsf{rbac}}(\sigma')$ est satisfait.
4. $R = \langle -_{\mathsf{PA}}, s, p, r \rangle$.
Par définition de $\tau_{\mathsf{rbac}}^{adm}$, $\sigma' = (m, user, \mathsf{UA}, \mathsf{PA} \setminus \{(p, r)\}, roles)$ et pour tout accès (s', o, x) appartenant à m, si $p = (x, o)$ alors (x, o) appartient à

$\mathsf{EP}(s', \mathsf{PA} \setminus \{(p,r)\}, roles)$. Par hypothèse, $\Omega_{\mathsf{rbac}}(\sigma)$ est vérifié, donc pour tout sujet s'', $roles(s'') \subseteq \mathsf{ER}(s'', \mathsf{UA})$ ce qui nous permet de déduire que σ' vérifie la propriété (1). De plus, par la même hypothèse, pour tout accès (s', o, x) appartenant à m, (x, o) appartient à $\mathsf{EP}(s', \mathsf{PA}, roles)$. Or, pour tout accès (s', o, x) appartenant à m, si $p = (x, o)$ alors (x, o) appartient à $\mathsf{EP}(s', \mathsf{PA} \setminus \{(p,r)\}, roles)$, et donc pour tout accès (s', o, x) appartenant à m, (x, o) appartient à $\mathsf{EP}(s', \mathsf{PA} \setminus \{(p,r)\}, roles)$, ce qui permet de déduire que σ' vérifie la propriété (2). Ainsi, σ' vérifie les propriétés (1) et (2), donc $\Omega_{\mathsf{rbac}}(\sigma')$ est satisfait.
5. $R = \langle +_{roles}, s, s', r \rangle$.
Par définition de $\tau^{adm}_{\mathsf{rbac}}$, $\sigma' = (m, user, \mathsf{UA}, \mathsf{PA}, roles[s' \leftarrow roles(s') \cup \{r\}])$ et r appartient à $\mathsf{ER}(s', \mathsf{UA})$. Par hypothèse, $\Omega_{\mathsf{rbac}}(\sigma)$ est vérifié, donc pour tout sujet s'', $roles(s'') \subseteq \mathsf{ER}(s'', \mathsf{UA})$. Or, r appartient à $\mathsf{ER}(s', \mathsf{UA})$, donc σ' vérifie la propriété (1). De plus, par la même hypothèse, pour tout accès (s''', o, x) appartenant à m, (x, o) appartient à $\mathsf{EP}(s''', \mathsf{PA}, roles)$. Or, pour tout sujet s'', $\mathsf{EP}(s'', \mathsf{PA}, roles) \subseteq \mathsf{EP}(s'', \mathsf{PA}, roles[s' \leftarrow roles(s') \cup \{r\}])$, donc σ' vérifie la propriété (2). Ainsi, σ' vérifie les propriétés (1) et (2), donc $\Omega_{\mathsf{rbac}}(\sigma')$ est satisfait.
6. $R = \langle -_{roles}, s, s', r \rangle$.
Par définition de $\tau^{adm}_{\mathsf{rbac}}$, $\sigma' = (m, user, \mathsf{UA}, \mathsf{PA}, roles[s' \leftarrow roles(s') \setminus \{r\}])$ et pour tout objet o, pour tout mode d'accès x, si (s', o, x) appartient à m alors (x, o) appartient à $\mathsf{EP}(s', \mathsf{PA}, roles[s' \leftarrow roles(s') \setminus \{r\}])$. Par hypothèse, $\Omega_{\mathsf{rbac}}(\sigma)$ est vérifié, donc pour tout sujet s'', $roles(s'') \subseteq \mathsf{ER}(s'', \mathsf{UA})$. Or, pour tout sujet s'', $roles[s' \leftarrow roles(s') \setminus \{r\}](s'') \subseteq roles(s'')$, donc σ' vérifie la propriété (1). De plus, par la même hypothèse, pour tout accès (s''', o', x') appartenant à m, (x', o') appartient à $\mathsf{EP}(s''', \mathsf{PA}, roles)$. Or, pour tout objet o et tout mode d'accès x, si (s', o, x) appartient à m alors (x, o) appartient à $\mathsf{EP}(s', \mathsf{PA}, roles[s' \leftarrow roles(s') \setminus \{r\}])$, et donc pour tout accès (s''', o', x') appartenant à m, (x', o') appartient à $\mathsf{EP}(s''', \mathsf{PA}, roles[s' \leftarrow roles(s') \setminus \{r\}])$, ce qui nous permet de déduire que σ' vérifie la propriété (2). Ainsi, σ' vérifie les propriétés (1) et (2), donc $\Omega_{\mathsf{rbac}}(\sigma')$ est satisfait.

• Montrons que $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}} \vdash \tau^{adm}_{\mathsf{rbac}}$.
Considérons deux états dans Σ_{rbac}, $\sigma_1 = (m_1, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles_1)$ et $\sigma_2 = (m_2, user_2, \mathsf{UA}_2, \mathsf{PA}_2, roles_2)$, et une requête R appartenant à $\mathcal{R}^{adm}$ tels que $\tau^{adm}_{\mathsf{rbac}}(R, \sigma_1) = (\mathsf{yes}, \sigma_2)$. Montrons que (R, σ_2) appartient à $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$. Ainsi, six cas sont possibles :
1. $R = \langle +_{\mathsf{UA}}, s, u, r \rangle$.
D'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, et étant donné que la réponse est yes, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1 \cup \{(u,r)\}, \mathsf{PA}_1, roles_1)$, et donc (u, r) appartient à UA_2, ce qui permet de conclure que $(\langle +_{\mathsf{UA}}, s, u, r \rangle, \sigma_2)$ appartient à $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$.
2. $R = \langle -_{\mathsf{UA}}, s, u, r \rangle$.
D'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, et étant donné que la réponse est yes, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1 \setminus \{(u,r)\}, \mathsf{PA}_1, roles_1)$, et donc (u, r) n'appartient pas à UA_2,

ce qui permet de conclure que $(\langle -_{\mathsf{UA}}, s, u, r\rangle, \sigma_2)$ appartient à $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$.
3. $R = \langle +_{\mathsf{PA}}, s, p, r\rangle$.
D'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, et étant donné que la réponse est yes, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1, \mathsf{PA}_1 \cup \{(p, r)\}, roles_1)$, et donc (p, r) appartient à PA_2, ce qui permet de conclure que $(\langle +_{\mathsf{PA}}, s, p, r\rangle, \sigma_2)$ appartient à $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$.
4. $R = \langle -_{\mathsf{PA}}, s, p, r\rangle$.
D'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, et étant donné que la réponse est yes, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1, \mathsf{PA}_1 \setminus \{(p, r)\}, roles_1)$, et donc (p, r) n'appartient pas à PA_2, ce qui permet de conclure que $(\langle -_{\mathsf{PA}}, s, p, r\rangle, \sigma_2)$ appartient à $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$.
5. $R = \langle +_{roles}, s, s', r\rangle$.
D'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, et étant donné que la réponse est yes, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles_1[s' \leftarrow roles_1(s') \cup \{r\}])$, et donc r appartient à $roles_2(s')$, ce qui permet de conclure que $(\langle +_{roles}, s, s', r\rangle, \sigma_2)$ appartient à $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$.
6. $R = \langle -_{roles}, s, s', r\rangle$.
D'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, et étant donné que la réponse est yes, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles_1[s' \leftarrow roles_1(s') \setminus \{r\}])$, et donc r n'appartient pas à $roles_2(s')$, ce qui permet de conclure que $(\langle -_{roles}, s, s', r\rangle, \sigma_2)$ appartient à $[\![\mathcal{R}_{\mathsf{rbac}}]\!]_{\Sigma_{\mathsf{rbac}}}$.

• Montrons que $\tau^{adm}_{\mathsf{rbac}}$ est $\mathcal{W}$-*conforme*.
Considérons deux états dans Σ_{rbac}, $\sigma_1 = (m_1, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles_1)$ et $\sigma_2 = (m_2, user_2, \mathsf{UA}_2, \mathsf{PA}_2, roles_2)$, une requête R appartenant à $\mathcal{R}^{adm}$ et une réponse d tels que $\tau^{adm}_{\mathsf{rbac}}(R, \sigma_1) = (d, \sigma_2)$. Si $d = \mathsf{no}$ alors, d'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, $\sigma_1 = \sigma_2$, et donc $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$ ce qui nous permet de conclure. Si $d = \mathsf{yes}$ alors six cas sont possibles :
1. $R = \langle +_{\mathsf{UA}}, s, u, r\rangle$.
D'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1 \cup \{(u, r)\}, \mathsf{PA}_1, roles_1)$. Nous devons montrer que $\mathcal{W}(\sigma_1) \subseteq \mathcal{W}(\sigma_2)$. Considérons E un ensemble d'accès tel que E appartient à $\mathcal{W}(\sigma_1)$. Par définition de $\mathcal{W}$, $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles_1))$ est vérifié, ainsi, pour tout sujet s', $roles_1(s') \subseteq \mathsf{ER}(s', \mathsf{UA}_1)$ et pour tout accès (s'', o, x) appartenant à $m_1 \cup E$, (x, o) appartient à $\mathsf{EP}(s'', \mathsf{PA}_1, roles_1)$. Or, pour tout sujet s', $\mathsf{ER}(s', \mathsf{UA}_1) \subseteq \mathsf{ER}(s', \mathsf{UA}_1 \cup \{(u, r)\})$, donc $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1 \cup \{(u, r)\}, \mathsf{PA}_1, roles_1))$ est satisfait ce qui nous permet de conclure que E appartient à $\mathcal{W}(\sigma_2)$.
2. $R = \langle -_{\mathsf{UA}}, s, u, r\rangle$.
D'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1 \setminus \{(u, r)\}, \mathsf{PA}_1, roles_1)$ et pour tout sujet s', si $user_1(s') = u$ alors $roles_1(s') \subseteq \mathsf{ER}(s', \mathsf{UA}_1 \setminus \{(u, r)\})$. Nous devons montrer que $\mathcal{W}(\sigma_2) \subseteq \mathcal{W}(\sigma_1)$. Considérons E un ensemble d'accès tel que E appartient à $\mathcal{W}(\sigma_2)$. Par définition de $\mathcal{W}$, $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1 \setminus \{(u, r)\}, \mathsf{PA}_1, roles_1))$ est vérifié, et donc, pour tout sujet s', $roles_1(s') \subseteq \mathsf{ER}(s', \mathsf{UA}_1 \setminus \{(u, r)\})$ et pour tout accès (s'', o, x) appartenant à $m_1 \cup E$, (x, o) appartient à $\mathsf{EP}(s'', \mathsf{PA}_1, roles_1)$. Or, pour tout sujet s', $\mathsf{ER}(s', \mathsf{UA}_1 \setminus \{(u, r)\}) \subseteq \mathsf{ER}(s', \mathsf{UA}_1)$, donc $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1, \mathsf{PA}_1,$

$roles_1$)) est satisfait ce qui nous permet de conclure que E appartient à $\mathcal{W}(\sigma_1)$.
3. $R = \langle +_{\mathsf{PA}}, s, p, r\rangle$.
D'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1, \mathsf{PA}_1 \cup \{(p,r)\}, roles_1)$. Nous devons montrer que $\mathcal{W}(\sigma_1) \subseteq \mathcal{W}(\sigma_2)$. Considérons E un ensemble d'accès tel que E appartient à $\mathcal{W}(\sigma_1)$. Par définition de $\mathcal{W}$, $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles_1))$ est vérifié, ainsi, pour tout sujet s', $roles_1(s') \subseteq \mathsf{ER}(s', \mathsf{UA}_1)$ et pour tout accès (s'', o, x) appartenant à $m_1 \cup E$, (x, o) appartient à $\mathsf{EP}(s'', \mathsf{PA}_1, roles_1)$. Or, pour tout sujet s', $\mathsf{EP}(s', \mathsf{PA}_1, roles_1) \subseteq \mathsf{EP}(s', \mathsf{PA}_1 \cup \{(p,r)\}, roles_1)$, donc $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1, \mathsf{PA}_1 \cup \{(p,r)\}, roles_1))$ est satisfait ce qui nous permet de conclure que E appartient à $\mathcal{W}(\sigma_2)$.
4. $R = \langle -_{\mathsf{PA}}, s, p, r\rangle$.
D'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1, \mathsf{PA}_1 \setminus \{(p,r)\}, roles_1)$ et pour tout accès (s', o, x) appartenant à m_1, si $p = (x, o)$ alors (x, o) appartient à $\mathsf{EP}(s', \mathsf{PA}_1 \setminus \{(p,r)\}, roles_1)$. Nous devons montrer que $\mathcal{W}(\sigma_2) \subseteq \mathcal{W}(\sigma_1)$. Considérons E un ensemble d'accès tel que E appartient à $\mathcal{W}(\sigma_2)$. Par définition de $\mathcal{W}$, $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1, \mathsf{PA}_1 \setminus \{(p,r)\}, roles_1))$ est vérifié, et donc, pour tout sujet s'', $roles_1(s'') \subseteq \mathsf{ER}(s'', \mathsf{UA}_1)$ et pour tout accès (s', o, x) appartenant à $m_1 \cup E$, (x, o) appartient à $\mathsf{EP}(s', \mathsf{PA}_1 \setminus \{(u,r)\}$, $roles_1)$. Or, pour tout sujet s'', $\mathsf{EP}(s'', \mathsf{PA}_1 \setminus \{(p,r)\}, roles_1) \subseteq \mathsf{EP}(s'', \mathsf{PA}_1$, $roles_1)$, donc $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles_1))$ est vérifié ce qui nous permet de conclure que E appartient à $\mathcal{W}(\sigma_1)$.
5. $R = \langle +_{roles}, s, s', r\rangle$.
Par définition de $\tau^{adm}_{\mathsf{rbac}}$, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles_1[s' \leftarrow roles_1(s') \cup \{r\}])$ et r appartient à $\mathsf{ER}(s', \mathsf{UA}_1)$. Nous devons montrer que $\mathcal{W}(\sigma_1) \subseteq \mathcal{W}(\sigma_2)$. Considérons E un ensemble d'accès tel que E appartient à $\mathcal{W}(\sigma_1)$. Par définition de $\mathcal{W}$, $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles_1))$ est vérifié, ainsi, pour tout sujet s'', $roles_1(s'') \subseteq \mathsf{ER}(s, \mathsf{UA}_1)$ et pour tout accès (s''', o, x) appartenant à $m_1 \cup E$, (x, o) appartient à $\mathsf{EP}(s''', \mathsf{PA}_1, roles_1)$. Or, pour tout sujet s'', $\mathsf{EP}(s'', \mathsf{PA}_1, roles_1) \subseteq \mathsf{EP}(s'', \mathsf{PA}_1, roles'_1)$ et r appartient à $\mathsf{ER}(s', \mathsf{UA}_1)$, donc $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles'_1))$ est satisfait ce qui nous permet de conclure que E appartient à $\mathcal{W}(\sigma_2)$.
6. $R = \langle -_{roles}, s, s', r\rangle$.
Posons $roles'_1 = roles_1[s' \leftarrow roles_1(s') \setminus \{r\}]$. D'après la définition de $\tau^{adm}_{\mathsf{rbac}}$, $\sigma_2 = (m_1, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles'_1)$, r appartient à $\mathsf{ER}(s', \mathsf{UA}_1)$ et pour tout objet o, pour tout mode d'accès x, si (s', o, x) appartient à m alors (x, o) appartient à $\mathsf{EP}(s', \mathsf{PA}_1, roles'_1)$. Nous devons montrer que $\mathcal{W}(\sigma_2) \subseteq \mathcal{W}(\sigma_1)$. Considérons E un ensemble d'accès tel que E appartient à $\mathcal{W}(\sigma_2)$. Par définition de $\mathcal{W}$, $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles'_1))$ est vérifié, et donc, pour tout sujet s'', $roles'_1(s'') \subseteq \mathsf{ER}(s'', \mathsf{UA}_1)$ et pour tout accès (s''', o', x') appartenant à $m_1 \cup E$, (x', o') appartient à $\mathsf{EP}(s''', \mathsf{PA}_1, roles'_1)$. Or, r appartient à $\mathsf{ER}(s', \mathsf{UA}_1)$ et pour tout sujet s'', $\mathsf{EP}(s'', \mathsf{PA}_1, roles'_1) \subseteq \mathsf{EP}(s'', \mathsf{PA}_1, roles_1)$, donc $\Omega_{\mathsf{rbac}}((m_1 \cup E, user_1, \mathsf{UA}_1, \mathsf{PA}_1, roles_1))$ est satisfait

ce qui nous permet de conclure que E appartient à $\mathcal{W}(\sigma_1)$. ◀

Les requêtes administratives permettent de modifier les fonctions de sécurité d'un état. La fonction de transition $\tau^{adm}_{\mathsf{rbac}}$ n'effectue aucun contrôle sur les sujets qui soumettent ces requêtes, elle se contente de s'assurer que les modifications souhaitées par le sujet ne conduisent pas à un état ne satisfaisant pas la politique RBAC96 (i.e. le prédicat Ω_{rbac}) et si c'est le cas, elle effectue les modifications. Ainsi, n'importe quel sujet peut changer les fonctions de sécurité d'un état, ce qui peut poser problème. Aussi, dans la pratique, il peut exister un rôle *Administrateur* étant le seul autorisé à effectuer des requêtes administratives. Ainsi, la fonction de transition doit vérifier que le sujet qui initie une requête administrative est au moins associé au rôle *Administrateur*. Une telle fonction de transition $\tau^{adm'}_{\mathsf{rbac}}$ est définie dans la table 3.

$$\tau^{adm'}_{\mathsf{rbac}}(\langle \__1, s, \__2, \__3\rangle, (m, user, \mathsf{UA}, \mathsf{PA}, roles))$$

$$= \begin{cases} \tau^{adm}_{\mathsf{rbac}}(\langle \__1, s, \__2, \__3\rangle, (m, user, \mathsf{UA}, \mathsf{PA}, roles)) \\ \quad \text{si} \quad Administrateur \in roles(s) \\ \\ (\mathsf{no}, (m, user, \mathsf{UA}, \mathsf{PA}, roles)) \text{ sinon} \end{cases}$$

TABLE 3 – Fonction de transition $\tau^{adm'}_{\mathsf{rbac}}$ pour RBAC96

4.2.3 Fonction de transition globale

Enfin, la table 4 contient la fonction de transition globale τ_{rbac} qui regroupe les fonctions $\tau^{acc}_{\mathsf{rbac}}$ et $\tau^{adm'}_{\mathsf{rbac}}$.

Bien sûr, cette fonction de transition permet d'obtenir une implantation correcte du modèle RBAC96.

Proposition 4 $\Sigma^{\mathsf{rbac}}_I \subseteq \Sigma_{\mathsf{rbac}_{|\Omega_{\mathsf{rbac}}}} \Rightarrow \mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \vdash (\tau_{\mathsf{rbac}}, \Sigma^{\mathsf{rbac}}_I)$

PREUVE. Conséquence immédiate des propositions 2 et 3. ◀

5 Comparaison de modèles de contrôle d'accès

Dans cette section, nous allons comparer le modèle RBAC96 avec le modèle de Bell et LaPadula. Pour cela, nous commençons par présenter la relation de comparaison que nous allons utiliser. Nous présentons ensuite une formalisation du modèle de Bell et LaPadula et nous terminons cette section en montrant que le modèle de Bell et LaPadula est plus restrictif que le modèle RBAC96.

$$\tau_{\mathsf{rbac}}(R,(m,user,\mathsf{UA},\mathsf{PA},roles))$$

$$= \begin{cases} \tau^{acc}_{\mathsf{rbac}}(R,(m,user,\mathsf{UA},\mathsf{PA},roles)) \\ \quad \text{si} \quad R \in \mathcal{R}^{acc} \\ \\ \tau^{adm'}_{\mathsf{rbac}}(R,(m,user,\mathsf{UA},\mathsf{PA},roles)) \\ \quad \text{si} \quad R \in \mathcal{R}^{adm} \\ \\ (\mathsf{no},(m,user,\mathsf{UA},\mathsf{PA},roles)) \text{ sinon} \end{cases}$$

TABLE 4 – Fonction de transition τ_{rbac} pour RBAC96

5.1 Préordre sur les modèles

Afin de pouvoir comparer des modèles de contrôle d'accès, nous commençons par exposer les définitions et les résultats introduits dans le cadre générique que nous utilisons pour définir les modèles.

Tout d'abord, les deux définitions suivantes caracatérisent les notions de restriction de politiques et de modèles de contrôle d'accès.

Définition 10 *Une politique* $\mathbb{P}_1[\rho] = (\mathcal{S},\mathcal{O},\mathcal{A},\Sigma_1,\Omega_1)$ *est une restriction de la politique* $\mathbb{P}_2[\rho] = (\mathcal{S},\mathcal{O},\mathcal{A},\Sigma_2,\Omega_2)$ *si et seulement si :*

$$\Sigma_1 \subseteq \Sigma_2 \wedge \forall \sigma \in \Sigma_1 \quad \Omega_1(\sigma) \Rightarrow \Omega_2(\sigma)$$

Définition 11 *Un modèle* $\mathbb{M}_1[\rho] = (\mathbb{P}_1[\rho], [\![\mathcal{R}]\!]_{\Sigma_1})$ *est une restriction du modèle* $\mathbb{M}_2[\rho] = (\mathbb{P}_2[\rho], [\![\mathcal{R}]\!]_{\Sigma_2})$ *si et seulement si* $\mathbb{P}_1[\rho]$ *est une restriction de* $\mathbb{P}_2[\rho]$ *et* $[\![\mathcal{R}]\!]_{\Sigma_1} \subseteq [\![\mathcal{R}]\!]_{\Sigma_2}$ *(sous réserve que l'ensemble de requêtes* $\mathcal{R}$ *et que la partition* $\mathcal{R} = \mathcal{R}^{\circledcirc} \cup \mathcal{R}^{\circledS} \cup \mathcal{R}^{\textcircled{1}}$ *soient les mêmes pour* $\mathbb{M}_1[\rho]$ *et* $\mathbb{M}_2[\rho]$*).*

L'approche suivie pour comparer les modèles de contrôle d'accès repose sur la simulation entre les implantations de ces modèles. Ainsi, on introduit les deux définitions suivantes.

Définition 12 (Simulation de fonctions de transition) *Soient deux fonctions de transition* $\tau_1 : \mathcal{R} \times \Sigma_1 \to \mathcal{D} \times \Sigma_1$ *et* $\tau_2 : \mathcal{R} \times \Sigma_2 \to \mathcal{D} \times \Sigma_2$*,* τ_2 *simule* τ_1*, ce que l'on note* $\tau_1 \overset{\kappa_\Sigma}{\sim} \tau_2$*, si et seulement si il existe une relation* $\kappa_\Sigma \subseteq \Sigma_1 \times \Sigma_2$ *telle que :*

$$\begin{array}{ll} & \forall \sigma_1,\sigma_1' \in \Sigma_1 \quad \forall \sigma_2 \in \Sigma_2 \quad \forall R \in \mathcal{R} \quad \forall d \in \mathcal{D} \\ & \quad ((\sigma_1,\sigma_2) \in \kappa_\Sigma \wedge \tau_1(R,\sigma_1) = (d,\sigma_1')) \\ \Rightarrow & \quad (\exists \sigma_2' \in \Sigma_2 \quad (\sigma_1',\sigma_2') \in \kappa_\Sigma \wedge \tau_2(R,\sigma_2) = (d,\sigma_2')) \end{array}$$

Définition 13 (Simulation d'implantations) *L'implantation* (τ_2, Σ_I^2) *simule* (τ_1, Σ_I^1)*, ce que l'on note* $(\tau_1, \Sigma_I^1) \overset{\kappa_\Sigma}{\rightarrowtail} (\tau_2, \Sigma_I^2)$*, si et seulement si il existe une relation* $\kappa_\Sigma \subseteq \Sigma_1 \times \Sigma_2$ *telle que*

$$\tau_1 \overset{\kappa_\Sigma}{\rightarrowtail} \tau_2 \wedge \forall \sigma_1 \in \Sigma_I^1 \quad \exists \sigma_2 \in \Sigma_I^2 \quad (\sigma_1, \sigma_2) \in \kappa_\Sigma$$

Nous verrons par la suite que les relations de simulation que nous allons considérer pour comparer des modèles doivent vérifier certaines propriétés que nous introduisons à présent.

Définition 14 *Etant donnés* $\mathbb{P}_1[\rho_1] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma_1, \Omega_1)$ *et* $\mathbb{P}_2[\rho_2] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma_2, \Omega_2)$ *deux politiques de contrôle d'accès,* $\mathbb{M}_1[\rho_1] = (\mathbb{P}_1[\rho_1], [\![\mathcal{R}]\!]_{\Sigma_1})$ *et* $\mathbb{M}_2[\rho_2] = (\mathbb{P}_2[\rho_2], [\![\mathcal{R}]\!]_{\Sigma_2})$ *deux modèles de contrôle d'accès, et une relation* $\kappa_\Sigma \subseteq \Sigma_1 \times \Sigma_2$*.*

1. κ_Σ *est dite* ι*-fonctionnelle si et seulement si :*

$$\begin{array}{l} \forall \sigma_1, \sigma_1' \in \Sigma_1 \quad \forall \sigma_2, \sigma_2' \in \Sigma_2 \\ \sigma_1 \equiv_{\iota_1} \sigma_1' \wedge (\sigma_1, \sigma_2) \in \kappa_\Sigma \wedge (\sigma_1', \sigma_2') \in \kappa_\Sigma \Rightarrow \sigma_2 \equiv_{\iota_2} \sigma_2' \end{array}$$

2. κ_Σ *est dite* ι*-injective si et seulement si :*

$$\begin{array}{l} \forall \sigma_1, \sigma_1' \in \Sigma_1 \quad \forall \sigma_2, \sigma_2' \in \Sigma_2 \\ \sigma_2 \equiv_{\iota_2} \sigma_2' \wedge (\sigma_1, \sigma_2) \in \kappa_\Sigma \wedge (\sigma_1', \sigma_2') \in \kappa_\Sigma \Rightarrow \sigma_1 \equiv_{\iota_1} \sigma_1' \end{array}$$

3. κ_Σ *est dite totale à gauche si et seulement si :*

$$\forall \sigma_1 \in \Sigma_1 \quad \exists \sigma_2 \in \Sigma_2 \quad (\sigma_1, \sigma_2) \in \kappa_\Sigma$$

4. κ_Σ *est dite* $\mathcal{W}$*-monotone si et seulement si :*

$$\begin{array}{l} \forall \sigma_1, \sigma_1' \in \Sigma_1 \quad \forall \sigma_2, \sigma_2' \in \Sigma_2 \\ (\mathcal{W}(\sigma_1) \subseteq \mathcal{W}(\sigma_1') \wedge (\sigma_1, \sigma_2) \in \kappa_\Sigma \wedge (\sigma_1', \sigma_2') \in \kappa_\Sigma) \\ \Rightarrow \mathcal{W}(\sigma_2) \subseteq \mathcal{W}(\sigma_2') \end{array}$$

5. κ_Σ *est dite* Ω*-préservante si et seulement si :*

$$\forall \sigma_1 \in \Sigma_1 \quad \forall \sigma_2 \in \Sigma_2 \quad ((\sigma_1, \sigma_2) \in \kappa_\Sigma \wedge \Omega_1(\sigma_1)) \Rightarrow \Omega_2(\sigma_2)$$

6. κ_Σ *est dite* $[\![\mathcal{R}]\!]_\Sigma$*-préservante si et seulement si :*

$$\begin{array}{l} \forall \sigma_1 \in \Sigma_1 \quad \forall \sigma_2 \in \Sigma_2 \quad \forall R \in \mathcal{R} \\ ((\sigma_1, \sigma_2) \in \kappa_\Sigma \wedge (R, \sigma_1) \in [\![\mathcal{R}]\!]_{\Sigma_1}) \Rightarrow (R, \sigma_2) \in [\![\mathcal{R}]\!]_{\Sigma_2} \end{array}$$

Nous pouvons à présent introduire les définitions de préordres sur les modèles de contrôle d'accès et sur les modèles réduits.

Définition 15 (Préordre sur les modèles) *Etant donnés deux modèles de contrôle d'accès* $\mathbb{M}_1[\rho_1] = (\mathbb{P}_1[\rho_1], [\![\mathcal{R}]\!]_{\Sigma_1})$ *et* $\mathbb{M}_2[\rho_2] = (\mathbb{P}_2[\rho_2], [\![\mathcal{R}]\!]_{\Sigma_2})$ *basés respectivement sur les politiques* $\mathbb{P}_1[\rho_1] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma_1, \Omega_1)$ *et* $\mathbb{P}_2[\rho_2] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma_2, \Omega_2)$, $\mathbb{M}_1[\rho_1]$ *est plus restrictif que* $\mathbb{M}_2[\rho_2]$, *ce que nous notons* $\mathbb{M}_1[\rho_1] \lessdot \mathbb{M}_2[\rho_2]$, *si et seulement si il existe une relation* $\kappa_\Sigma \subseteq \Sigma_1 \times \Sigma_2$ *totale à gauche,* ι*-fonctionnelle et* ι*-injective telle que :*

$$\begin{array}{ll} & \forall \tau_1 : \mathcal{R} \times \Sigma_1 \to \mathcal{D} \times \Sigma_1 \quad \forall \Sigma_I^1 \subseteq \Sigma_1 \\ & \mathbb{M}_1[\rho_1] \vdash (\tau_1, \Sigma_I^1) \wedge \equiv_{\iota_1} \vdash \tau_1 \\ \Rightarrow & \exists \tau_2 : \mathcal{R} \times \Sigma_2 \to \mathcal{D} \times \Sigma_2 \quad \exists \Sigma_I^2 \subseteq \Sigma_2 \\ & \mathbb{M}_2[\rho_2] \vdash (\tau_2, \Sigma_I^2) \wedge \equiv_{\iota_2} \vdash \tau_2 \wedge (\tau_1, \Sigma_I^1) \stackrel{\kappa_\Sigma}{\rightarrow} (\tau_2, \Sigma_I^2) \end{array}$$

Définition 16 (Préordre sur les modèles réduits) *Etant donnés deux modèles* $\mathbb{M}_1[\rho_1] = (\mathbb{P}_1[\rho_1], [\![\mathcal{R}]\!]_{\Sigma_1})$ *et* $\mathbb{M}_2[\rho_2] = (\mathbb{P}_2[\rho_2], [\![\mathcal{R}]\!]_{\Sigma_2})$ *respectivement basés sur les politiques* $\mathbb{P}_1[\rho_1] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma_1, \Omega_1)$ *et* $\mathbb{P}_2[\rho_2] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma_2, \Omega_2)$, *tels que* $\mathbb{M}_1[\rho_1]$ *et* $\mathbb{M}_2[\rho_2]$ *soient des modèles réduits construits à partir de* $\equiv_\iota$, $\mathbb{M}_1[\rho_1]$ *est plus restrictif que* $\mathbb{M}_2[\rho_2]$, *ce que nous notons* $\mathbb{M}_1[\rho_1] \trianglelefteq \mathbb{M}_2[\rho_2]$, *si et seulement si il existe une relation* $\kappa_\Sigma \subseteq \Sigma_1 \times \Sigma_2$ *totale à gauche, injective et fonctionnelle telle que :*

$$\begin{array}{ll} & \forall \tau_1 : \mathcal{R} \times \Sigma_1 \to \mathcal{D} \times \Sigma_1 \quad \forall \Sigma_I^1 \subseteq \Sigma_1 \\ & \mathbb{M}_1[\rho_1] \vdash (\tau_1, \Sigma_I^1) \\ \Rightarrow & \exists \tau_2 : \mathcal{R} \times \Sigma_2 \to \mathcal{D} \times \Sigma_2 \quad \exists \Sigma_I^2 \subseteq \Sigma_2 \\ & \mathbb{M}_2[\rho_2] \vdash (\tau_2, \Sigma_I^2) \wedge (\tau_1, \Sigma_I^1) \stackrel{\kappa_\Sigma}{\rightarrow} (\tau_2, \Sigma_I^2) \end{array}$$

Enfin, nous introduisons le lemme et les théorèmes suivants, dont les preuves sont fournies dans [**?**].

Lemme 7 *Si* $\mathbb{M}_1[\rho]$ *est une restriction de* $\mathbb{M}_2[\rho]$ *alors* $\mathbb{M}_1[\rho] \lessdot \mathbb{M}_2[\rho]$.

Théorème 1 $\mathbb{M}_1[\rho_1] \lessdot \mathbb{M}_2[\rho_2] \Leftrightarrow \mathbb{M}_1^\sharp[\rho_1] \trianglelefteq \mathbb{M}_2^\sharp[\rho_2]$.

Théorème 2 *Etant donnés deux modèles de contrôle d'accès* $\mathbb{M}_1[\rho_1] = (\mathbb{P}_1[\rho_1], [\![\mathcal{R}]\!]_{\Sigma_1})$ *et* $\mathbb{M}_2[\rho_2] = (\mathbb{P}_2[\rho_2], [\![\mathcal{R}]\!]_{\Sigma_2})$ *basés respectivement sur les politiques* $\mathbb{P}_1[\rho_1] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma_1, \Omega_1)$ *et* $\mathbb{P}_2[\rho_2] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma_2, \Omega_2)$, *et* $\kappa_\Sigma \subseteq \Sigma_1 \times \Sigma_2$ *une relation totale à gauche,* $\mathcal{W}$*-monotone,* ι*-fonctionnelle et* ι*-injective. Si* κ_Σ *est* Ω*-préservante et* $[\![\mathcal{R}]\!]_\Sigma$*-préservante alors* $\mathbb{M}_1[\rho_1] \lessdot \mathbb{M}_2[\rho_2]$.

5.2 Modèle de Bell et LaPadula

Nous introduisons ici le modèle de Bell et LaPadula tel qu'il est formalisé dans [**?**].

Le modèle de Bell et LaPadula contraint les accès possibles en utilisant un *treillis de niveaux de sécurité* associé aux sujets et aux objets, noté $\rho_{\mathsf{blp}} = (\mathcal{L}, \preceq, \sqcup, \sqcap)$. Dans ce contexte, un état appartenant à Σ_{blp} est un triplet $\sigma =$

(m, f_s, f_o) où $\Lambda(\sigma) = m$ est l'ensemble des accès courants et $\Upsilon(\sigma) = (f_s, f_o)$ correspond aux fonctions de sécurité. La fonction de sécurité $f_s : \mathcal{S} \to \mathcal{L}$ (resp. $f_o : \mathcal{O} \to \mathcal{L}$) définit les niveaux de sécurité associés aux sujets (resp. objets). Les fonctions f_s et f_o sont parfois appelées des vecteurs de classification.

La *politique de sécurité de Bell et LaPadula* est spécifiée par un prédicat Ω_{blp}. Etant donné un état $\sigma = (m, f_s, f_o)$ appartenant à Σ_{blp}, $\Omega_{\mathsf{blp}}(\sigma)$ est satisfait si et seulement si les deux propriétés suivantes sont vérifiées :

$$\forall s \in \mathcal{S} \quad \forall o \in \mathcal{O} \quad (s, o, \mathsf{read}) \in m \Rightarrow f_o(o) \preceq f_s(s) \tag{9}$$

$$\begin{array}{l} \forall s \in \mathcal{S} \quad \forall o_1, o_2 \in \mathcal{O} \\ (s, o_1, \mathsf{read}) \in m \wedge (s, o_2, \mathsf{write}) \in m \Rightarrow f_o(o_1) \preceq f_o(o_2) \end{array} \tag{10}$$

On note $\mathbb{P}_{\mathsf{blp}}[\rho_{\mathsf{blp}}] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma_{\mathsf{blp}}, \Omega_{\mathsf{blp}})$ la politique ainsi définie. De plus, on considère le langage de requêtes défini en (3), muni de la sémantique des requêtes définie en (5). Ainsi, on note $\mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}] = (\mathbb{P}_{\mathsf{blp}}[\rho_{\mathsf{blp}}], [\![\mathcal{R}^{acc}]\!]_{\Sigma_{\mathsf{blp}}})$ le modèle de contrôle d'accès de Bell et LaPadula.

5.3 Comparaison de deux modèles

Nous allons maintenant comparer les modèles de Bell et LaPadula et de RBAC96. Comparer deux modèles de contrôle d'accès consiste principalement à définir une interprétation du formalisme d'un modèle dans le formalisme de l'autre. Cette interprétation est présentée de manière plus ou moins formelle dans la littérature. Par exemple, [**?**] présente de manière intuitive comment le modèle RBAC96 peut être utilisé pour mettre en oeuvre une variante de la politique de Bell et LaPadula pour le système Multics [**?**]. Nous fournissons ici une approche différente et complètement formalisée pour comparer le modèle RBAC96 avec le modèle mathématique original de Bell et LaPadula [**?**]. Pour cela, nous suivons la méthodologie utilisée dans [**?**] pour comparer le modèle de la Muraille de Chine avec celui de Bell et LaPadula.

L'approche suivie ici consiste, dans un premier temps, à construire un ensemble d'utilisateurs et une hiérarchie de rôles $\rho_{\mathsf{rblp}} = \kappa_\rho(\mathcal{S}, \rho_{\mathsf{blp}})$ à partir d'un treillis de niveaux de sécurité ρ_{blp} et de l'ensemble des sujets $\mathcal{S}$. Puis, dans un deuxième temps, nous introduisons un modèle $\mathbb{M}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}]$ correspondant à une interprétation à base de rôles du modèle de Bell et LaPadula. Enfin, étant donné que nous souhaitons montrer que le modèle $\mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}]$ de Bell et LaPadula est plus restrictif que le modèle $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$ de RBAC96, nous allons montrer que $\mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}] \lessdot \mathbb{M}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}]$, ainsi que $\mathbb{M}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}] \lessdot \mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$, ce qui nous permettra de conclure que $\mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}] \lessdot \mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$. Nous montrons également que le modèle de RBAC96 n'est pas plus restrictif que celui de Bell et LaPadula, $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \not\lessdot \mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}]$.

5.3.1 Traduction du paramètre de sécurité

A partir de l'ensemble des sujets $\mathcal{S}$ et du treillis des niveaux de sécurité $\rho_{\mathsf{blp}} = (\mathcal{L}, \preceq, \sqcup, \sqcap)$ correspondant au paramètre de sécurité du modèle de Bell et LaPadula, nous construisons le paramètre de sécurité $\rho_{\mathsf{rblp}} = \kappa_\rho(\mathcal{S}, \rho_{\mathsf{blp}})$, définissant un ensemble de sujets et un ensemble de rôles muni d'une relation d'ordre partiel, de la manière suivante. Tout d'abord, l'ensemble des sujets, qui nous sera utile par la suite pour faire le lien avec le paramètre de sécurité du modèle RBAC96, est inchangé. De plus, à chaque niveau de sécurité l du treillis $\mathcal{L}$ correspond un rôle de même nom et l'ordre partiel sur les rôles est le même que celui associé au treillis. Ainsi, la fonction κ_ρ permettant cette construction est la fonction identité (en omettant $\sqcup$ et $\sqcap$ étant donné que nous avons uniquement besoin d'un ordre partiel), c'est-à-dire :

$$\rho_{\mathsf{rblp}} = (\mathcal{S}, \mathcal{L}, \preceq)$$

5.3.2 Traduction de la notion d'état

La traduction de l'ensemble des états Σ_{blp}, de la forme $\sigma = (m, f_s, f_o)$, se fait en construisant un ensemble d'états Σ_{rblp} exprimant les états de Σ_{blp} avec les notations introduites pour les états appartenant à Σ_{rbac}. Ainsi, un état appartenant à Σ_{rblp} contient un ensemble d'accès m et des fonctions de sécurité définies de la manière suivante.

La fonction *user* permet d'associer un utilisateur à un sujet dans le modèle RBAC96. Sa traduction est la fonction identité

$$user_{\mathsf{rblp}} : \mathcal{S} \to \mathcal{S}$$

étant donné que la notion d'utilisateurs n'existe pas dans le modèle de Bell et LaPadula.

La relation $\mathsf{UA}_{\mathsf{rblp}}$ correspond à l'ensemble

$$\mathsf{UA}_{\mathsf{rblp}} = \{(s, f_s(s)) \mid s \in \mathcal{S}\}$$

Cet ensemble met en relation un seul rôle pour chaque sujet car un unique niveau de sécurité leur est attribué.

La relation $\mathsf{PA}_{\mathsf{rblp}}$ est définie par l'ensemble

$$\mathsf{PA}_{\mathsf{rblp}} = \{((\mathsf{read}, o), f_o(o)), ((\mathsf{write}, o), f_o(o)), ((\mathsf{write}, o), \bot) \mid o \in \mathcal{O}\}$$

Les deux premiers éléments traduisent le fait que tous les objets peuvent être accédés en lecture et en écriture selon leur niveau de sécurité. Le troisième élément ajoute une restriction stipulant que tous les objets sont de plus associés au niveau de sécurité le plus faible (noté $\bot$), ce qui permet à tous les sujets d'écrire dans n'importe quel objet. Bien entendu, comme nous le verrons par la suite, cette possibilité est contrainte afin que la politique

de sécurité de Bell et LaPadula soit respectée (un sujet peut écrire dans n'importe quel objet s'il ne lit aucun objet).

Enfin, pour tout sujet s, $roles_{\mathsf{rblp}}(s)$ est égal au singleton :

$$roles_{\mathsf{rblp}}(s) = \{f_s(s)\}$$

ce qui caractérise le fait qu'un sujet active toujours le rôle correspondant à son niveau de sécurité.

Ainsi, un état σ appartenant à Σ_{rblp} issu de la traduction d'un état de Σ_{blp} s'écrit comme un 5-uplet :

$$\sigma = (m, user_{\mathsf{rblp}}, \mathsf{UA}_{\mathsf{rblp}}, \mathsf{PA}_{\mathsf{rblp}}, roles_{\mathsf{rblp}})$$

Bien sûr, l'ensemble Σ_{rblp} est inclus dans Σ_{rbac}.

5.3.3 Traduction du prédicat de sécurité

Nous pouvons à présent introduire le prédicat de sécurité Ω_{rblp}. Etant donné un état $\sigma = (m, user_{\mathsf{rblp}}, \mathsf{UA}_{\mathsf{rblp}}, \mathsf{PA}_{\mathsf{rblp}}, roles_{\mathsf{rblp}})$ appartenant à Σ_{rblp}, le prédicat Ω_{rblp} est vérifié si et seulement si les deux propriétés suivantes sont satisfaites.

$$\begin{array}{l} \forall s \in \mathcal{S} \quad \forall o \in \mathcal{O} \\ (s, o, \mathsf{read}) \in m \Rightarrow \\ \left(\forall r, r' \in \mathcal{L} \left(\begin{array}{ll} & r \in roles_{\mathsf{rblp}}(s) \\ \wedge & ((\mathsf{read}, o), r') \in \mathsf{PA}_{\mathsf{rblp}} \end{array} \right) \Rightarrow r' \preceq r \right) \end{array} \tag{11}$$

$$\begin{array}{l} \forall s \in \mathcal{S} \quad \forall o_1, o_2 \in \mathcal{O} \\ (s, o_1, \mathsf{read}) \in m \wedge (s, o_2, \mathsf{write}) \in m \Rightarrow \\ \left(\begin{array}{l} \forall r, r' \in \mathcal{L} \\ \left(\begin{array}{ll} & ((\mathsf{read}, o_1), r) \in \mathsf{PA}_{\mathsf{rblp}} \\ \wedge & r' = \bigsqcup \{ r'' \in \mathcal{L} \mid ((\mathsf{write}, o_2), r'') \in \mathsf{PA}_{\mathsf{rblp}} \} \end{array} \right) \Rightarrow r \preceq r' \end{array} \right) \end{array} \tag{12}$$

La propriété (11) énonce le fait que si un sujet s lit un objet o alors le rôle r' associé à la permission en lecture de o est inférieur au rôle r associé à s. Quant à la propriété (12), elle décrit le fait que si un sujet s lit un objet o_1 et écrit dans un objet o_2 alors le rôle r associé à la permission en lecture de o_1 est inférieur au supremum des rôles associés à la permission en écriture de o_2 (c'est-à-dire au rôle r' correspondant au niveau de sécurité de o_2 étant donné que le supremum est calculé sur un ensemble contenant au plus deux rôles, r' et $\bot$). Ainsi, la propriété (11) (resp. (12)) est la traduction de la propriété (9) (resp. (10)) dans le formalisme RBAC.

On définit ainsi la politique :

$$\mathbb{P}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}] = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \Sigma_{\mathsf{rblp}}, \Omega_{\mathsf{rblp}})$$

correspondant à une interprétation à base de rôles du modèle de Bell et LaPadula. Nous montrons que cette politique est compacte et correspond à une restriction de la politique RBAC96.

Proposition 5 $\mathbb{P}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}]$ *est compacte.*

PREUVE. Considérons deux états σ_1 et σ_2 appartenant à Σ_{rblp} tels que $\Omega_{\mathsf{rblp}}(\sigma_1)$ soit satisfait, $\Lambda(\sigma_2) \subseteq \Lambda(\sigma_1)$ et $\Upsilon(\sigma_1) = \Upsilon(\sigma_2)$. Nous voulons montrer que $\Omega_{\mathsf{rblp}}(\sigma_2)$ est satisfait, c'est-à-dire que σ_2 vérifie les propriétés (11) et (12). Par hypothèse, pour tout sujet s et tout objet o, si l'accès (s, o, read) appartient à $\Lambda(\sigma_2)$ alors il appartient à $\Lambda(\sigma_1)$ et les états σ_1 et σ_2 ont les mêmes fonctions de sécurité. Or, étant donné que $\Omega_{\mathsf{rblp}}(\sigma_1)$ est satisfait, pour tout rôles r et r', si r appartient à $roles_{\mathsf{rblp}}(s)$ et si le couple $((\mathsf{read}, o), r')$ appartient à $\mathsf{PA}_{\mathsf{rblp}}$ alors $r' \preceq r$. On en déduit que σ_2 vérifie la propriété (11). De même, pour tout sujet s et tout objets o_1 et o_2, si les accès (s, o_1, read) et (s, o_2, write) appartiennent à $\Lambda(\sigma_2)$ alors ils appartiennent à $\Lambda(\sigma_1)$ et les états σ_1 et σ_2 ont les mêmes fonctions de sécurité. Or, étant donné que $\Omega_{\mathsf{rblp}}(\sigma_1)$ est satisfait, pour tout rôles r et r', si le couple $((\mathsf{read}, o_1), r)$ appartient à $\mathsf{PA}_{\mathsf{rblp}}$ et si r' est le supremum de tout rôle r'' tel que $((\mathsf{write}, o_2), r'')$ appartient à $\mathsf{PA}_{\mathsf{rblp}}$ alors $r \preceq r'$. On en déduit que σ_2 vérifie la propriété (12), ce qui nous permet de conclure que $\Omega_{\mathsf{rblp}}(\sigma_2)$ est satisfait. ◄

Proposition 6 $\mathbb{P}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}]$ *est une restriction de* $\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$.

PREUVE. Par construction de Σ_{rblp}, il est clair que $\Sigma_{\mathsf{rblp}} \subseteq \Sigma_{\mathsf{rbac}}$. Montrons à présent que pour tout état σ appartenant à Σ_{rblp}, si σ satisfait le prédicat Ω_{rblp} alors il satisfait également le prédicat Ω_{rbac}. Considérons un état σ appartenant à Σ_{rblp} tel que $\Omega_{\mathsf{rblp}}(\sigma)$ soit vérifié, montrons que σ vérifie les propriétés (1) et (2). Etant donné que par construction, pour tout sujet s, $roles_{\mathsf{rblp}}(s) = \{r\}$ tel que (s, r) appartient à $\mathsf{UA}_{\mathsf{rblp}}$, on en déduit que $roles_{\mathsf{rblp}}(s) \subseteq \mathsf{ER}(s, \mathsf{UA}_{\mathsf{rblp}})$, et donc que σ vérifie la propriété (1). D'autre part, pour tout sujet s et tout objet o, si $\Lambda(\sigma)$ contient des accès de la forme (s, o, read) alors par hypothèse, d'après la propriété (11), le rôle associé à la permission (read, o) est inférieur (selon $\preceq$) au rôle associé au sujet s, ce qui nous permet de déduire que la permission (read, o) appartient à $\mathsf{EP}(s, \mathsf{PA}_{\mathsf{rblp}}, roles_{\mathsf{rblp}})$. De plus, si $\Lambda(\sigma)$ contient des accès de la forme (s, o, write) alors étant donné que par construction le rôle $\perp$ est associé à la permission (write, o), il est inférieur (selon $\preceq$) au rôle associé au sujet s, ce qui nous permet de déduire que la permission (write, o) appartient à $\mathsf{EP}(s, \mathsf{PA}_{\mathsf{rblp}}, roles_{\mathsf{rblp}})$, et donc que σ vérifie la propriété (2). Ainsi, σ vérifie les propriétés (1) et (2), ce qui nous permet de conclure que $\Omega_{\mathsf{rbac}}(\sigma)$ est satisfait. ◄

On définit à présent le modèle :

$$\mathbb{M}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}] = (\mathbb{P}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}], [\![\mathcal{R}^{acc}]\!]_{\Sigma_{\mathsf{rblp}}})$$

en considérant le langage de requêtes défini en (3) muni de la sémantique donnée en (5) avec x appartenant à $\{\mathsf{read}, \mathsf{write}\}$. De même que pour les politiques, le modèle $\mathbb{M}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}]$ est une restriction du modèle $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$.

Proposition 7 *$\mathbb{M}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}]$ est une restriction de $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$.*

PREUVE. Immédiat d'après la proposition 6 et le fait que la sémantique des requêtes est la même pour les deux modèles. ◀

5.3.4 Relation de simulation

La traduction des états appartenant à Σ_{blp} vers des états appartenant à Σ_{rblp} permet de définir une relation de simulation $\kappa_\Sigma \subseteq \Sigma_{\mathsf{blp}} \times \Sigma_{\mathsf{rblp}}$ formalisant cette constuction :

$$\begin{array}{l} \forall \sigma_1 = (m_1, f_s, f_o) \in \Sigma_{\mathsf{blp}} \\ \forall \sigma_2 = (m_2, user_{\mathsf{rblp}}, \mathsf{UA}_{\mathsf{rblp}}, \mathsf{PA}_{\mathsf{rblp}}, roles_{\mathsf{rblp}}) \in \Sigma_{\mathsf{rblp}} \end{array}$$

$$\begin{array}{rl} & (\sigma_1, \sigma_2) \in \kappa_\Sigma \\ \Leftrightarrow & \left(\begin{array}{ll} & m_1 = m_2 \\ \wedge & \forall s \in \mathcal{S} \quad f_s(s) = l \Leftrightarrow \\ & \mathsf{UA}_{\mathsf{rblp}} = \{(s,l)\} \wedge roles_{\mathsf{rblp}}(s) = \{l\} \\ \wedge & \forall o \in \mathcal{O} \quad f_o(o) = l \Leftrightarrow \\ & \mathsf{PA}_{\mathsf{rblp}} = \{((\mathsf{read}, o), l), ((\mathsf{write}, o), l), ((\mathsf{write}, o), \bot)\} \end{array} \right) \end{array}$$

Afin de prouver que κ_Σ est totale à gauche, *ι-fonctionnelle*, *ι-injective* et *$\mathcal{W}$-monotone*, nous montrons tout d'abord le lemme suivant.

Lemme 8

$$\forall \sigma_1 \in \Sigma_{\mathsf{blp}} \quad \forall \sigma_2 \in \Sigma_{\mathsf{rblp}} \quad (\sigma_1, \sigma_2) \in \kappa_\Sigma \Rightarrow \left(\begin{array}{ll} & \mathcal{W}_\emptyset(\sigma_1) = \mathcal{W}_\emptyset(\sigma_2) \\ \wedge & \mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2) \\ \wedge & \Lambda(\sigma_1) = \Lambda(\sigma_2) \end{array} \right)$$

PREUVE. Considérons deux états $\sigma_1 = (m_1, f_s, f_o)$ et $\sigma_2 = (m_2, user_{\mathsf{rblp}}, \mathsf{UA}_{\mathsf{rblp}}, \mathsf{PA}_{\mathsf{rblp}}, roles_{\mathsf{rblp}})$ tels que (σ_1, σ_2) appartient à κ_Σ. Par construction de κ_Σ, $m_1 = m_2$ c'est-à-dire $\Lambda(\sigma_1) = \Lambda(\sigma_2)$. Afin de démontrer que $\mathcal{W}_\emptyset(\sigma_1) = \mathcal{W}_\emptyset(\sigma_2)$, nous allons montrer l'inclusion dans les deux sens.
1. $\mathcal{W}_\emptyset(\sigma_1) \subseteq \mathcal{W}_\emptyset(\sigma_2)$.
Considérons E un ensemble d'accès tel que E appartient à $\mathcal{W}_\emptyset(\sigma_1)$. Posons $\sigma_1' = (E, f_s, f_o)$ et $\sigma_2' = (E, user_{\mathsf{rblp}}, \mathsf{UA}_{\mathsf{rblp}}, \mathsf{PA}_{\mathsf{rblp}}, roles_{\mathsf{rblp}})$. Par définition de $\mathcal{W}$, $\Omega_{\mathsf{blp}}(\sigma_1')$ est satisfait et on veut montrer que E appartient à $\mathcal{W}_\emptyset(\sigma_2)$, c'est-à-dire que $\Omega_{\mathsf{rblp}}(\sigma_2')$ est vérifié. Pour cela, nous montrons que σ_2' vérifie les propriétés (11) et (12). Etant donné que $\Omega_{\mathsf{blp}}(\sigma_1')$ est satisfait, l'état σ_1' vérifie les propriétés (9) et (10). La propriété (9) stipule qu'un sujet ne peut

lire un objet que si le niveau de sécurité de cet objet est inférieur au niveau de sécurité du sujet. Or, dans la propriété (11), le rôle r correspond au niveau de sécurité du sujet et le rôle r' à celui de l'objet. On en déduit que l'état σ_2' vérifie la propriété (11). D'autre part, la propriété (10) énonce le fait que si un sujet lit un objet o_1 et écrit dans un objet o_2 alors le niveau de sécurité associé à l'objet o_1 est inférieur à celui associé à l'objet o_2. Or, dans la propriété (12), le rôle r correspond au niveau de sécurité de o_1 et le rôle r' correspond au niveau de sécurité de o_2 étant donné qu'il est obtenu à partir du supremum entre le rôle correspondant au niveau de sécurité de o_2 et $\bot$. On en déduit que l'état σ_2' vérifie la propriété (12), ce qui nous permet de conclure que $\Omega_{\mathsf{rblp}}(\sigma_2')$ est satisfait.
2. $\mathcal{W}_\emptyset(\sigma_2) \subseteq \mathcal{W}_\emptyset(\sigma_1)$.
Un raisonnement similaire au point 1. permet de conclure.
Nous avons jusqu'à présent montré que si (σ_1, σ_2) appartient à κ_Σ alors $\Lambda(\sigma_1) = \Lambda(\sigma_2)$ et $\mathcal{W}_\emptyset(\sigma_1) = \mathcal{W}_\emptyset(\sigma_2)$. D'après le lemme 4, on en déduit que $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$, ce qui nous permet de conclure. ◀

Proposition 8 *κ_Σ est totale à gauche, ι-fonctionnelle, ι-injective et $\mathcal{W}$-monotone.*

Preuve. Nous commençons par montrer que la relation κ_Σ est *ι-fonctionnelle*, *ι-injective* et *$\mathcal{W}$-monotone* en utilisant le lemme 8. Puis nous montrons qu'elle est totale à gauche.
1. κ_Σ est *ι-fonctionnelle.*
Considérons deux états σ_1 et σ_1' appartenant à Σ_{blp}, et deux états σ_2 et σ_2' appartenant à Σ_{rblp} tels que $\sigma_1 \equiv_{\iota_{\mathsf{blp}}} \sigma_1'$, (σ_1, σ_2) appartient à κ_Σ et (σ_1', σ_2') appartient à κ_Σ. Nous voulons montrer que $\sigma_2 \equiv_{\iota_{\mathsf{rblp}}} \sigma_2'$. Or d'après la définition de $\equiv_\iota$, $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_1')$, $\mathcal{W}_\emptyset(\sigma_1) = \mathcal{W}_\emptyset(\sigma_1')$ et $\sigma_1 \equiv_{\mathcal{R}^{acc}} \sigma_1'$, et d'après le lemme 8, $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$, $\mathcal{W}_\emptyset(\sigma_1) = \mathcal{W}_\emptyset(\sigma_2)$, $\Lambda(\sigma_1) = \Lambda(\sigma_2)$, $\mathcal{W}(\sigma_1') = \mathcal{W}(\sigma_2')$, $\mathcal{W}_\emptyset(\sigma_1') = \mathcal{W}_\emptyset(\sigma_2')$ et $\Lambda(\sigma_1') = \Lambda(\sigma_2')$, ce qui nous permet de conlure que $\mathcal{W}(\sigma_2) = \mathcal{W}(\sigma_2')$, $\mathcal{W}_\emptyset(\sigma_2) = \mathcal{W}_\emptyset(\sigma_2')$ et $\sigma_2 \equiv_{\mathcal{R}^{acc}} \sigma_2'$ (car on considère la même sémantique des requêtes pour les états appartenant à Σ_{blp} et Σ_{rblp}), c'est-à-dire que $\sigma_2 \equiv_{\iota_{\mathsf{rblp}}} \sigma_2'$.
2. κ_Σ est ι-injective.
Un raisonnement similaire au point 1. permet de conclure.
3. κ_Σ est *$\mathcal{W}$-monotone.*
Considérons deux états σ_1 et σ_1' appartenant à Σ_{blp}, et deux états σ_2 et σ_2' appartenant à Σ_{rblp} tels que $\mathcal{W}(\sigma_1) \subseteq \mathcal{W}(\sigma_1')$, (σ_1, σ_2) appartient à κ_Σ et (σ_1', σ_2') appartient à κ_Σ. Nous voulons montrer que $\mathcal{W}(\sigma_2) \subseteq \mathcal{W}(\sigma_2')$. Or, d'après le lemme 8, $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$ et $\mathcal{W}(\sigma_1') = \mathcal{W}(\sigma_2')$, ce qui nous permet de conclure que $\mathcal{W}(\sigma_2) \subseteq \mathcal{W}(\sigma_2')$.
4. κ_Σ est totale à gauche.
Considérons un état $\sigma_1 = (m_1, f_s^1, f_o^1)$ appartenant à Σ_{blp} et un état $\sigma_2 = (m_2, user_2, \mathsf{UA}_2, \mathsf{PA}_2, roles_2)$, où $m_2 = m_1$, $user_2(s) = s$ pour tout sujet

s, $\mathsf{UA}_2 = \{(s, f_s^1(s)) \mid s \in \mathcal{S}\}$, $\mathsf{PA}_2 = \{((\mathsf{read}, o), f_o^1(o)), ((\mathsf{write}, o), f_o^1(o)), ((\mathsf{write}, o), \bot) \mid o \in \mathcal{O}\}$ et $roles_2(s) = \{f_s^1(s)\}$ pour tout sujet s. Par construction, σ_2 appartient à Σ_{rblp} et (σ_1, σ_2) appartient à κ_Σ. La relation κ_Σ est par conséquent totale à gauche. ◀

5.3.5 Comparaison

Nous pouvons à présent montrer que le modèle $\mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}]$ est plus restrictif que le modèle $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$.

Proposition 9 $\mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}] \lessdot \mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$

Preuve. D'après la proposition 7, le modèle $\mathbb{M}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}]$ est une restriction du modèle $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$. Or, d'après le lemme 7, on en déduit que $\mathbb{M}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}] \lessdot \mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$. Ainsi, pour montrer que $\mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}] \lessdot \mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$, il nous reste à montrer que $\mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}] \lessdot \mathbb{M}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}]$. Pour cela, nous allons utiliser le théorème 2. Ainsi, nous commençons par montrer que κ_Σ est Ω-*préservante*. Considérons un état σ_1 appartenant à Σ_{blp} et un état σ_2 appartenant à Σ_{rblp} tels que (σ_1, σ_2) appartient à κ_Σ et $\Omega_{\mathsf{blp}}(\sigma_1)$ soit satisfait. Montrons que $\Omega_{\mathsf{rblp}}(\sigma_2)$ est vérifié. D'après le lemme 1, il vient $\mathcal{W}(\sigma_1) \neq \emptyset$. De plus, d'après le lemme 8 et étant donné que (σ_1, σ_2) appartient à κ_Σ, on en déduit que $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2)$, et donc que $\mathcal{W}(\sigma_2) \neq \emptyset$. D'après la proposition 5, $\mathbb{P}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}]$ est compacte, ce qui nous permet de conclure, d'après le lemme 2, que $\Omega_{\mathsf{rblp}}(\sigma_2)$ est satisfait. De manière similaire, nous montrons que κ_Σ est $[\![\mathcal{R}]\!]_\Sigma$-*préservante*. Considérons un état σ_1 appartenant à Σ_{blp}, un état σ_2 appartenant à Σ_{rblp} et une requête R appartenant $\mathcal{R}^{acc}$ tels que (σ_1, σ_2) appartient à κ_Σ et (R, σ_1) appartient à $[\![\mathcal{R}^{acc}]\!]_{\Sigma_{\mathsf{blp}}}$. Montrons que (R, σ_2) appartient à $[\![\mathcal{R}^{acc}]\!]_{\Sigma_{\mathsf{rblp}}}$. D'après le lemme 8 et étant donné que (σ_1, σ_2) appartient à κ_Σ, on en déduit que $\Lambda(\sigma_1) = \Lambda(\sigma_2)$. Or, étant donné que les sémantiques des requêtes pour les états appartenant à Σ_{blp} et Σ_{rblp} sont les mêmes (elles correspondent à celle définie en (5) avec x appartenant à $\{\mathsf{read}, \mathsf{write}\}$), on en déduit que (R, σ_2) appartient à $[\![\mathcal{R}^{acc}]\!]_{\Sigma_{\mathsf{rblp}}}$. Ainsi, La relation κ_Σ est Ω-*préservante* et $[\![\mathcal{R}]\!]_\Sigma$-*préservante*, et comme elle est totale à gauche, ι-*fonctionnelle*, ι-*injective* et $\mathcal{W}$-*monotone* d'après la proposition 8, nous pouvons conclure d'après le théorème 2, que $\mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}] \lessdot \mathbb{M}_{\mathsf{rblp}}[\rho_{\mathsf{rblp}}]$ et donc que $\mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}] \lessdot \mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rblp}}]$. ◀

De plus, nous montrons que le modèle $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}]$ n'est pas plus restrictif que le modèle $\mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}]$.

Proposition 10 $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \not\lessdot \mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}]$

Preuve. Plus précisément, nous montrons

$$\neg(\forall \rho_{\mathsf{rbac}} \quad \exists \rho_{\mathsf{blp}} \quad \mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \lessdot \mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}])$$

Pour ce faire, nous montrons que $\mathbb{M}^{\sharp}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \not\trianglelefteq \mathbb{M}^{\sharp}_{\mathsf{blp}}[\rho_{\mathsf{blp}}]$. Ainsi, nous montrons qu'il n'existe pas de relation de simulation $\kappa_{\Sigma} \subseteq \hat{e}(\Sigma_{\mathsf{rbac}}) \times \hat{e}(\Sigma_{\mathsf{blp}})$ qui soit à la fois totale à gauche, injective et fonctionnelle. Considérons les ensembles suivants : $\mathcal{S} = \{s\}$, $\mathcal{O} = \{o\}$, $\mathcal{A} = \{\mathsf{write}\}$, $\mathsf{U} = \{u\}$, $\mathsf{R} = \{r\}$, et $\mathcal{R} = \{\langle +, s, o, \mathsf{write}\rangle, \langle -, s, o, \mathsf{write}\rangle\}$. En supposant que le sujet s est toujours actif (c'est-à-dire que $user(s)$ est toujours égal à u), on en déduit que :

$$\Sigma_{\mathsf{rbac}} = \left\{ \begin{array}{lcl}
\sigma_1 & = & \left(\begin{array}{l} \emptyset, user(s) = u, \mathsf{UA} = \emptyset, \\ \mathsf{PA} = \emptyset, roles(s) = \emptyset \end{array} \right), \\
\sigma_2 & = & \left(\begin{array}{l} \emptyset, user(s) = u, \mathsf{UA} = \{(u,r)\}, \\ \mathsf{PA} = \emptyset, roles(s) = \emptyset \end{array} \right), \\
\sigma_3 & = & \left(\begin{array}{l} \emptyset, user(s) = u, \mathsf{UA} = \emptyset, \\ \mathsf{PA} = \{((\mathsf{write}, o), r)\}, roles(s) = \emptyset \end{array} \right), \\
\sigma_4 & = & \left(\begin{array}{l} \emptyset, user(s) = u, \mathsf{UA} = \emptyset, \\ \mathsf{PA} = \emptyset, roles(s) = \{r\} \end{array} \right), \\
\sigma_5 & = & \left(\begin{array}{l} \emptyset, user(s) = u, \mathsf{UA} = \{(u,r)\}, \\ \mathsf{PA} = \{((\mathsf{write}, o), r)\}, roles(s) = \emptyset \end{array} \right), \\
\sigma_6 & = & \left(\begin{array}{l} \emptyset, user(s) = u, \mathsf{UA} = \{(u,r)\}, \\ \mathsf{PA} = \emptyset, roles(s) = \{r\} \end{array} \right), \\
\sigma_7 & = & \left(\begin{array}{l} \emptyset, user(s) = u, \mathsf{UA} = \emptyset, \\ \mathsf{PA} = \{((\mathsf{write}, o), r)\}, roles(s) = \{r\} \end{array} \right), \\
\sigma_8 & = & \left(\begin{array}{l} \emptyset, user(s) = u, \mathsf{UA} = \{(u,r)\}, \\ \mathsf{PA} = \{((\mathsf{write}, o), r)\}, roles(s) = \{r\} \end{array} \right), \\
\sigma_9 & = & \left(\begin{array}{l} \{(s, o, \mathsf{write})\}, user(s) = u, \mathsf{UA} = \emptyset, \\ \mathsf{PA} = \emptyset, roles(s) = \emptyset \end{array} \right), \\
\sigma_{10} & = & \left(\begin{array}{l} \{(s, o, \mathsf{write})\}, user(s) = u, \mathsf{UA} = \{(u,r)\}, \\ \mathsf{PA} = \emptyset, roles(s) = \emptyset \end{array} \right), \\
\sigma_{11} & = & \left(\begin{array}{l} \{(s, o, \mathsf{write})\}, user(s) = u, \mathsf{UA} = \emptyset, \\ \mathsf{PA} = \{((\mathsf{write}, o), r)\}, roles(s) = \emptyset \end{array} \right), \\
\sigma_{12} & = & \left(\begin{array}{l} \{(s, o, \mathsf{write})\}, user(s) = u, \mathsf{UA} = \emptyset, \\ \mathsf{PA} = \emptyset, roles(s) = \{r\} \end{array} \right), \\
\sigma_{13} & = & \left(\begin{array}{l} \{(s, o, \mathsf{write})\}, user(s) = u, \mathsf{UA} = \{(u,r)\}, \\ \mathsf{PA} = \{((\mathsf{write}, o), r)\}, roles(s) = \emptyset \end{array} \right), \\
\sigma_{14} & = & \left(\begin{array}{l} \{(s, o, \mathsf{write})\}, user(s) = u, \mathsf{UA} = \{(u,r)\}, \\ \mathsf{PA} = \emptyset, roles(s) = \{r\} \end{array} \right), \\
\sigma_{15} & = & \left(\begin{array}{l} \{(s, o, \mathsf{write})\}, user(s) = u, \mathsf{UA} = \emptyset, \\ \mathsf{PA} = \{((\mathsf{write}, o), r)\}, roles(s) = \{r\} \end{array} \right), \\
\sigma_{16} & = & \left(\begin{array}{l} \{(s, o, \mathsf{write})\}, user(s) = u, \mathsf{UA} = \{(u,r)\}, \\ \mathsf{PA} = \{((\mathsf{write}, o), r)\}, roles(s) = \{r\} \end{array} \right)
\end{array} \right\}$$

Notons tout d'abord que pour tout état σ appartenant à Σ_{rbac}, $\mathcal{W}(\sigma) = \mathcal{W}_{\emptyset}(\sigma)$. Or, par définition de $[\![\mathcal{R}^{acc}]\!]_{\Sigma_{\mathsf{rbac}}}$, $\sigma_1 \equiv_{\mathcal{R}} \sigma_2 \equiv_{\mathcal{R}} \sigma_3 \equiv_{\mathcal{R}} \sigma_4 \equiv_{\mathcal{R}} \sigma_5 \equiv_{\mathcal{R}} \sigma_6 \equiv_{\mathcal{R}} \sigma_7 \equiv_{\mathcal{R}} \sigma_8$ et $\sigma_9 \equiv_{\mathcal{R}} \sigma_{10} \equiv_{\mathcal{R}} \sigma_{11} \equiv_{\mathcal{R}} \sigma_{12} \equiv_{\mathcal{R}} \sigma_{13} \equiv_{\mathcal{R}} \sigma_{14} \equiv_{\mathcal{R}} \sigma_{15} \equiv_{\mathcal{R}} \sigma_{16}$. De plus, par définition de $\mathcal{W}$, $\mathcal{W}(\sigma_1) = \mathcal{W}(\sigma_2) = \mathcal{W}(\sigma_3) = \mathcal{W}(\sigma_4) =$

$\mathcal{W}(\sigma_5) = \mathcal{W}(\sigma_6) = \mathcal{W}(\sigma_7) = \{\emptyset\}$, $\mathcal{W}(\sigma_9) = \mathcal{W}(\sigma_{10}) = \mathcal{W}(\sigma_{11}) = \mathcal{W}(\sigma_{12}) = \mathcal{W}(\sigma_{13}) = \mathcal{W}(\sigma_{14}) = \mathcal{W}(\sigma_{15}) = \emptyset$ et $\mathcal{W}(\sigma_8) = \mathcal{W}(\sigma_{16}) = \{\emptyset, \{(s, o, \mathsf{write})\}\}$. On en déduit que $\sigma_1 \equiv_\iota \sigma_2 \equiv_\iota \sigma_3 \equiv_\iota \sigma_4 \equiv_\iota \sigma_5 \equiv_\iota \sigma_6 \equiv_\iota \sigma_7$ et $\sigma_9 \equiv_\iota \sigma_{10} \equiv_\iota \sigma_{11} \equiv_\iota \sigma_{12} \equiv_\iota \sigma_{13} \equiv_\iota \sigma_{14} \equiv_\iota \sigma_{15}$. Considérons $e : \Sigma_{\mathsf{rbac}} \to \Sigma_{\mathsf{rbac}}$ la fonction de projection associée à $\equiv_\iota$ définie par :

$$\forall \sigma \in \Sigma_{\mathsf{rbac}} \quad e(\sigma) = \begin{cases} \sigma_1 & \text{si} \quad \sigma = \sigma_1 \vee \sigma = \sigma_2 \vee \sigma = \sigma_3 \vee \sigma = \sigma_4 \vee \\ & \qquad \sigma = \sigma_5 \vee \sigma = \sigma_6 \vee \sigma = \sigma_7 \\ \sigma_9 & \text{si} \quad \sigma = \sigma_9 \vee \sigma = \sigma_{10} \vee \sigma = \sigma_{11} \vee \sigma = \sigma_{12} \vee \\ & \qquad \sigma = \sigma_{13} \vee \sigma = \sigma_{14} \vee \sigma = \sigma_{15} \\ \sigma_8 & \text{si} \quad \sigma = \sigma_8 \\ \sigma_{16} & \text{si} \quad \sigma = \sigma_{16} \end{cases}$$

Cette définition nous permet de conclure que $\mathbf{card}(\hat{e}(\Sigma_{\mathsf{rbac}})) = 4$. Montrons à présent par l'absurde que pour les ensembles $\mathcal{S}$, $\mathcal{O}$, $\mathcal{A}$ et $\mathcal{R}$ définis précédemment, $\mathbf{card}(\hat{e}(\Sigma_{\mathsf{blp}})) \leq 2$. En effet, supposons que le cardinal de $\hat{e}(\Sigma_{\mathsf{blp}})$ soit strictement supérieur à 2. Ainsi, il existe trois états σ_1, σ_2 et σ_3 appartenant à $\hat{e}(\Sigma_{\mathsf{blp}})$ tels que $\sigma_1 \not\equiv_\iota \sigma_2$, $\sigma_1 \not\equiv_\iota \sigma_3$ et $\sigma_2 \not\equiv_\iota \sigma_3$. Or, par définition des ensembles $\mathcal{S}$, $\mathcal{O}$, $\mathcal{A}$ et $\mathcal{R}$, pour tout état σ appartenant à $\hat{e}(\Sigma_{\mathsf{blp}})$, $\Lambda(\sigma) = \emptyset$ ou $\Lambda(\sigma) = \{(s, o, \mathsf{write})\}$. Or, selon la politique de Bell et LaPadula, il est toujours possible d'ajouter un accès en écriture aux accès courants d'un état si l'on effectue aucun accès en lecture. Ainsi, pour tout état σ appartenant à $\hat{e}(\Sigma_{\mathsf{blp}})$, $\mathcal{W}(\sigma) = \{\emptyset, \{(s, o, \mathsf{write})\}\}$, et donc $\sigma_1 \equiv_{\mathcal{W}} \sigma_2 \equiv_{\mathcal{W}} \sigma_3$. Or, étant donné qu'ici encore, pour tout état σ appartenant à Σ_{blp}, $\mathcal{W}(\sigma) = \mathcal{W}_\emptyset(\sigma)$, on en déduit que $\sigma_1 \equiv_{\mathcal{W}_\emptyset} \sigma_2 \equiv_{\mathcal{W}_\emptyset} \sigma_3$. De plus, puisque pour tout état σ appartenant à $\hat{e}(\Sigma_{\mathsf{blp}})$, $\Lambda(\sigma) = \emptyset$ ou $\Lambda(\sigma) = \{(s, o, \mathsf{write})\}$, au moins deux états parmi σ_1, σ_2 et σ_3 ont le même ensemble d'accès courants. Supposons que ces deux états soient σ_1 et σ_2 (le raisonnement est similaire si ces deux états sont σ_1 et σ_3 ou σ_2 et σ_3). Etant donné que $\Lambda(\sigma_1) = \Lambda(\sigma_2)$, on en déduit que $\sigma_1 \equiv_{\mathcal{R}} \sigma_2$. Or, $\sigma_1 \equiv_{\mathcal{W}} \sigma_2$ et $\sigma_1 \equiv_{\mathcal{W}_\emptyset} \sigma_2$, et donc $\sigma_1 \equiv_\iota \sigma_2$, ce qui est en contradiction avec nos hypothèses. Ainsi, nous concluons que $\mathbf{card}(\hat{e}(\Sigma_{\mathsf{blp}})) \leq 2$. Nous avons montré que $\mathbf{card}(\hat{e}(\Sigma_{\mathsf{rbac}})) > \mathbf{card}(\hat{e}(\Sigma_{\mathsf{blp}}))$ ce qui implique qu'il est impossible de construire une relation totale à gauche, fonctionnelle et injective entre $\hat{e}(\Sigma_{\mathsf{rbac}})$ et $\hat{e}(\Sigma_{\mathsf{blp}})$. Nous pouvons donc conclure que $\mathbb{M}^\sharp_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \not\trianglelefteq \mathbb{M}^\sharp_{\mathsf{blp}}[\rho_{\mathsf{blp}}]$ et donc, d'après le théorème 1, que $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}] \not\trianglelefteq \mathbb{M}_{\mathsf{blp}}[\rho_{\mathsf{blp}}]$. ◀

6 Exemple

Dans cette section, nous illustrons l'utilisation de la formalisation du modèle RBAC96 décrite dans les sections précédentes sur l'exemple concret d'une partie d'un hôpital.

Entités L'ensemble des sujets est générique :

$$\mathcal{S}_e = \left\{ \begin{array}{ccccccccccccc} s_1 & , & s_2 & , & s_3 & , & s_4 & , & s_5 & , & s_6 & , \\ s_7 & , & s_8 & , & s_9 & , & s_{10} & , & s_{11} & , & s_{12} & \end{array} \right\}$$

L'ensemble des objets est le suivant :

$$\mathcal{O}_e = \left\{ \begin{array}{ccc} DossierMedecin\_i & , & DossierPatient\_i \\ Ordonnance\_i & , & CompteRenduOp\_i \\ Radio\_i & | & i \in \{1,2,3\} \end{array} \right\}$$

Accès L'ensemble des permissions est :

$$\mathsf{P}_e = \left\{ \begin{array}{ccccc} (& x\_DossierMedecin & , & DossierMedecin\_i &), \\ (& x\_PartieAdminPatient & , & DossierPatient\_i &), \\ (& x\_PartieMedPatient & , & DossierPatient\_i &), \\ (& x\_Ordonnance & , & Ordonnance\_i &), \\ (& x\_CompteRenduOp & , & CompteRenduOp\_i &), \\ (& x\_Radio & , & Radio\_i &) \\ | & x \in \{Activer, Consulter, Modifier\} & \wedge & i \in \{1,2,3\} & \end{array} \right\}$$

Paramètre de sécurité Les personnes travaillant à l'hôpital et/ou étant des patients composent l'ensemble des utilisateurs :

$$\mathsf{U}_e = \left\{ \begin{array}{ccccccccccc} Alice & , & Bob & , & Charly & , & Dalia & , & Estel & , \\ Franck & , & Gregory & , & Helen & , & Isabel & , & John & \end{array} \right\}$$

L'ensemble des rôles est :

$$\mathsf{R}_e = \left\{ \begin{array}{ccc} Directeur & , & Secretaire & , \\ ChirurgienChef & , & Chirurgien & , \\ Radiologiste & , & RadiologisteAssistant & , \\ Specialiste & , & Generaliste & , \\ Infirmiere & , & SecretaireMedicale & , \\ Patient & & & \end{array} \right\}$$

L'ordre partiel $\leq_{\mathsf{R}_e}$ sur les rôles de R_e est représenté à la figure 1.

Etats On considère les attributions suivantes :

$$\begin{array}{cccccccccc} user_e(s_1) & = & Alice & ; & user_e(s_2) & = & Bob & ; \\ user_e(s_3) & = & Charly & ; & user_e(s_4) & = & Dalia & ; \\ user_e(s_5) & = & Dalia & ; & user_e(s_6) & = & Estel & ; \\ user_e(s_7) & = & Franck & ; & user_e(s_8) & = & Gregory & ; \\ user_e(s_9) & = & Helen & ; & user_e(s_{10}) & = & Isabel & ; \\ user_e(s_{11}) & = & John & ; & user_e(s_{12}) & = & John & ; \end{array}$$

On voit ici que plusieurs sujets peuvent correspondre à un même utilisateur.

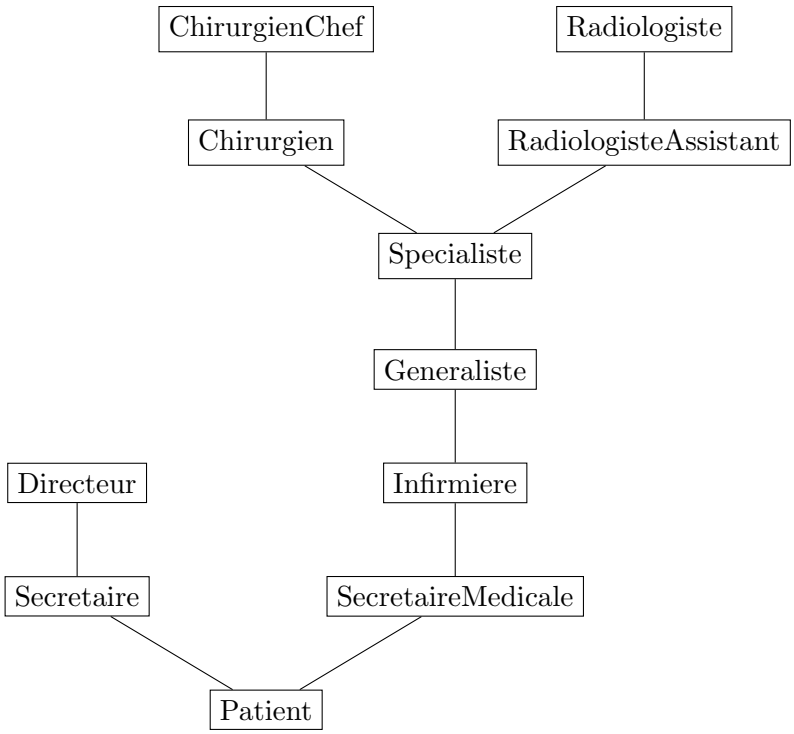


FIGURE 1 – Ordre partiel $\leq_{\mathsf{R}_e}$

$$\mathsf{UA}_e = \left\{ \begin{array}{lcc} (& Alice & , \quad Directeur), \\ (& Bob & , \quad ChirurgienChef), \\ (& Charly & , \quad Radiologiste), \\ (& Dalia & , \quad Chirurgien), \\ (& Estel & , \quad SecretaireMedicale), \\ (& Franck & , \quad Chirurgien), \\ (& Gregory & , \quad Generaliste), \\ (& Helen & , \quad Secretaire), \\ (& Isabel & , \quad Infirmiere), \\ (& John & , \quad RadiologisteAssistant) \end{array} \right\}$$

$$\mathsf{PA}_e = \left\{ \begin{array}{lcr} ((Activer\_DossierMedecin & , & DossierMedecin\_i), \\ \quad Directeur), & & \\ ((\{Consulter, Modifier\}\_DossierMedecin & , & DossierMedecin\_i), \\ \quad Secretaire), & & \\ ((Consulter\_DossierMedecin & , & DossierMedecin\_i), \\ \quad Generaliste), & & \\ ((\{Activer, Modifier\}\_PartieAdminPatient & , & DossierPatient\_i), \\ \quad SecretaireMedicale), & & \\ ((Consulter\_PartieAdminPatient & , & DossierPatient\_i), \\ \quad Patient), & & \\ ((\{Activer, Modifier\}\_PartieMedPatient & , & DossierPatient\_i), \\ \quad Generaliste), & & \\ ((Consulter\_PartieMedPatient & , & DossierPatient\_i), \\ \quad Patient), & & \\ ((\{Activer, Modifier\}\_Ordonnance & , & Ordonnance\_i), \\ \quad Generaliste), & & \\ ((Consulter\_Ordonnance & , & Ordonnance\_i), \\ \quad Patient), & & \\ ((\{Activer, Modifier\}\_CompteRenduOp & , & CompteRenduOp\_i), \\ \quad Chirurgien), & & \\ ((Consulter\_CompteRenduOp & , & CompteRenduOp\_i), \\ \quad Patient), & & \\ ((Activer\_Radio & , & Radio\_i), \\ \quad RadiologisteAssistant), & & \\ ((Consulter\_Radio & , & Radio\_i), \\ \quad Patient) & | & i \in \{1, 2, 3\} \end{array} \right\}$$

$$\begin{array}{lcl} roles_e(s_1) & = & \{ \quad Directeur \quad \} \ ; \\ roles_e(s_2) & = & \{ \quad ChirurgienChef \quad , \quad Chirurgien \quad \} \ ; \\ roles_e(s_3) & = & \{ \quad Radiologiste \quad \} \ ; \\ roles_e(s_4) & = & \{ \quad Chirurgien \quad \} \ ; \\ roles_e(s_5) & = & \{ \quad Chirurgien \quad , \quad Generaliste \quad \} \ ; \\ roles_e(s_6) & = & \{ \quad SecretaireMedicale \quad \} \ ; \\ roles_e(s_7) & = & \{ \quad Chirurgien \quad \} \ ; \\ roles_e(s_8) & = & \{ \quad Generaliste \quad \} \ ; \\ roles_e(s_9) & = & \{ \quad Secretaire \quad \} \ ; \\ roles_e(s_{10}) & = & \{ \quad Infirmiere \quad , \quad Patient \quad \} \ ; \\ roles_e(s_{11}) & = & \{ \quad RadiologisteAssistant \quad \} \ ; \\ roles_e(s_{12}) & = & \{ \quad Generaliste \quad \} \ ; \end{array}$$

Prédicat de sécurité Pour chaque session s_i, l'ensemble dénoté par la fonction ER est :

$$\mathsf{ER}_e(s_1, \mathsf{UA}) = \left\{ \begin{array}{ccc} Directeur & , & Secretaire \\ Patient & & \end{array} , \right\};$$

$$\mathsf{ER}_e(s_2, \mathsf{UA}) = \left\{ \begin{array}{ccc} ChirurgienChef & , & Chirurgien \\ Specialiste & , & Generaliste \\ Infirmiere & , & SecretaireMedicale \\ Patient & & \end{array} , \right\};$$

$$\mathsf{ER}_e(s_3, \mathsf{UA}) = \left\{ \begin{array}{ccc} Radiologiste & , & RadiologisteAssistant \\ Specialiste & , & Generaliste \\ Infirmiere & , & SecretaireMedicale \\ Patient & & \end{array} , \right\};$$

$$\mathsf{ER}_e(s_4, \mathsf{UA}) = \left\{ \begin{array}{ccc} Chirurgien & , & Specialiste \\ Generaliste & , & Infirmiere \\ SecretaireMedicale & , & Patient \end{array} , \right\};$$

$$\mathsf{ER}_e(s_5, \mathsf{UA}) = \left\{ \begin{array}{ccc} Chirurgien & , & Specialiste \\ Generaliste & , & Infirmiere \\ SecretaireMedicale & , & Patient \end{array} , \right\};$$

$$\mathsf{ER}_e(s_6, \mathsf{UA}) = \{ \; SecretaireMedicale \quad , \quad Patient \quad \};$$

$$\mathsf{ER}_e(s_7, \mathsf{UA}) = \left\{ \begin{array}{ccc} Chirurgien & , & Specialiste \\ Generaliste & , & Infirmiere \\ SecretaireMedicale & , & Patient \end{array} , \right\};$$

$$\mathsf{ER}_e(s_8, \mathsf{UA}) = \left\{ \begin{array}{ccc} Generaliste & , & Infirmiere \\ SecretaireMedicale & , & Patient \end{array} , \right\};$$

$$\mathsf{ER}_e(s_9, \mathsf{UA}) = \{ \; Secretaire \quad , \quad Patient \quad \};$$

$$\mathsf{ER}_e(s_{10}, \mathsf{UA}) = \left\{ \begin{array}{ccc} Infirmiere & , & SecretaireMedicale \\ Patient & & \end{array} , \right\};$$

$$\mathsf{ER}_e(s_{11}, \mathsf{UA}) = \left\{ \begin{array}{ccc} RadiologisteAssistant & , & Specialiste \\ Generaliste & , & Infirmiere \\ SecretaireMedicale & , & Patient \end{array} , \right\};$$

$$\mathsf{ER}_e(s_{12}, \mathsf{UA}) = \left\{ \begin{array}{ccc} RadiologisteAssistant & , & Specialiste \\ Generaliste & , & Infirmiere \\ SecretaireMedicale & , & Patient \end{array} , \right\};$$

Par souci de place, seuls trois exemples de la fonction EP sont donnés.

$$\mathsf{EP}_e(s_1, \mathsf{PA}, roles) =$$
$$\left\{ \begin{array}{lccc} (& x\_DossierMedecin & , & DossierMedecin\_i &), \\ (& Consulter\_PartieAdminPatient & , & DossierPatient\_i &), \\ (& Consulter\_PartieMedPatient & , & DossierPatient\_i &), \\ (& Consulter\_Ordonnance & , & Ordonnance\_i &), \\ (& Consulter\_CompteRenduOp & , & CompteRenduOp\_i &), \\ (& Consulter\_Radio & , & Radio\_i &) \\ | & x \in \{Activer, Consulter, Modifier\} & \wedge & i \in \{1,2,3\} & \end{array} \right\};$$

$$\mathsf{EP}_e(s_2, \mathsf{PA}, roles) =$$
$$\left\{ \begin{array}{lccc} (& Consulter\_DossierMedecin & , & DossierMedecin\_i &), \\ (& x\_PartieAdminPatient & , & DossierPatient\_i &), \\ (& x\_PartieMedPatient & , & DossierPatient\_i &), \\ (& x\_Ordonnance & , & Ordonnance\_i &), \\ (& x\_CompteRenduOp & , & CompteRenduOp\_i &), \\ (& Consulter\_Radio & , & Radio\_i &) \\ | & x \in \{Activer, Consulter, Modifier\} & \wedge & i \in \{1,2,3\} & \end{array} \right\};$$

$$\mathsf{EP}_e(s_{12}, \mathsf{PA}, roles) =$$
$$\left\{ \begin{array}{lccc} (& Consulter\_DossierMedecin & , & DossierMedecin\_i &), \\ (& x\_PartieAdminPatient & , & DossierPatient\_i &), \\ (& x\_PartieMedPatient & , & DossierPatient\_i &), \\ (& x\_Ordonnance & , & Ordonnance\_i &), \\ (& Consulter\_CompteRenduOp & , & CompteRenduOp\_i &), \\ (& Consulter\_Radio & , & Radio\_i &) \\ | & x \in \{Activer, Consulter, Modifier\} & \wedge & i \in \{1,2,3\} & \end{array} \right\};$$

Requêtes Un exemple de requête est :

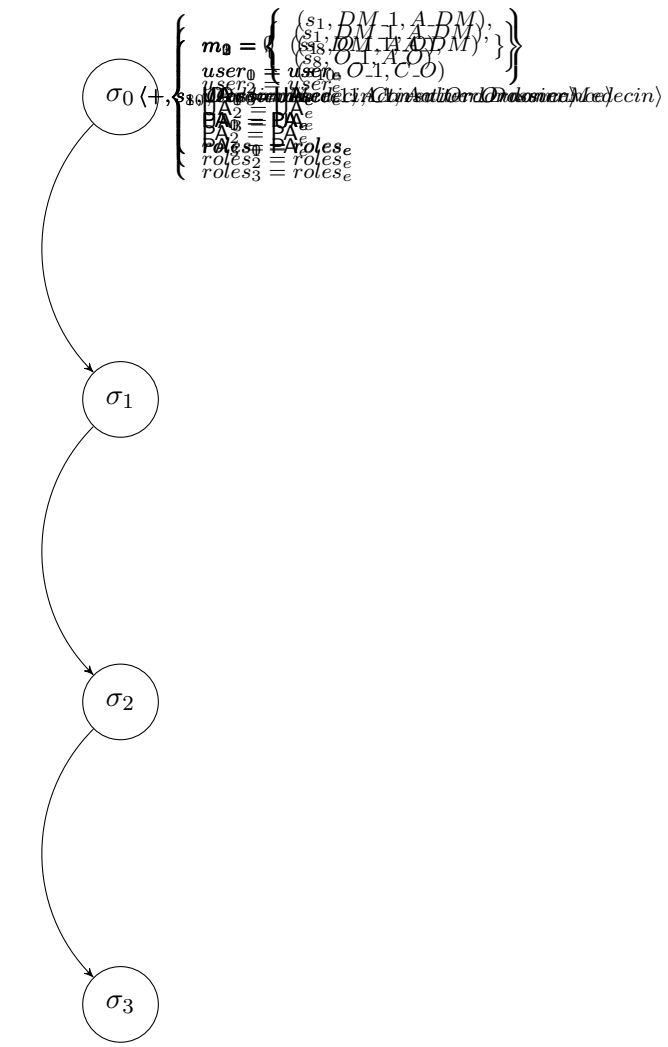
$$\langle +, s_8, Ordonnance\_1, Activer\_Ordonnance \rangle$$

Implantations La figure 2 représente, sous forme de machine abstraite à états, le système en « fonctionnement », c'est-à-dire qu'elle montre l'évolution des états en fonction des requêtes.

7 Implantations Focal

L'implantation réalisée dans l'atelier Focal [**?**] comporte deux parties principales. La première met en œuvre une partie de la formalisation du modèle RBAC96 exposée dans ce document tandis que la deuxième concrétise cette formalisation par raffinements successifs jusqu'à aboutir à une implantation de l'exemple de la section précédente.

Avant de présenter la partie de l'implantation qui correspond à la formalisation de RBAC96, nous introduisons l'essentiel de l'architecture réalisée par Charles Morisset sur laquelle repose notre développement. Cette architecture implante certains éléments du cadre vus précédemment et a été



$DM = DossierMedecin$; $A\_DM = Activer\_DossierMedecin$;
$O = Ordonnance$; $A\_O = Activer\_Ordonnance$;
$C\_O = Consulter\_Ordonnance$;

FIGURE 2 – Exemple d'évolution du système

complétée de propriétés que nous avons utilisées pour effectuer les preuves de correction de la fonction de transition et de la sémantique des requêtes du modèle RBAC96 en collaboration avec Florian Brecht, stagiaire dans l'équipe SPI.

7.1 L'atelier Focal

L'environnement Focal fournit un langage construit sur OCaml et Coq pour spécifier, programmer et prouver des unités de bibliothèques. Il permet donc à la fois d'écrire des programmes et de prouver certaines propriétés qu'ils vérifient. Les unités de bibliothèques sont traduites par le compilateur Focal vers du code source OCaml pour aboutir à des programmes exécutables, vers un format appelé FocDoc générant des fichiers XML de documentation, et vers du code source Coq afin de vérifier les preuves. L'environnement Focal propose un langage d'expression de propriétés qui peuvent être vérifiées et/ou prouvées soit à l'aide de Coq, soit à l'aide de Zenon, un outil permettant de construire une preuve en s'appuyant sur la structure particulière des objets spécifiés dans l'environnement Focal. Cette preuve est ensuite automatiquement vérifiée par Coq. En outre, le langage de Focal permet un développement modulaire par passage progressif de la spécification à l'implantation grâce aux traits objets dont il dispose (héritage, redéfinition, instanciation, ...).

En Focal, les structures sont représentées par des espèces. L'implantation des bibliothèques peut être effectuée étape par étape grâce à la notion d'héritage qui relie les espèces entres elles. Une espèce définit une structure par un ensemble de méthodes. A l'intérieur d'une espèce, certaines méthodes peuvent être uniquement déclarées (donc non définies), ce qui permet de les manipuler sans connaître leur implantation réelle. La notion d'héritage permet de définir une espèce à partir d'autres espèces préalablement définies. Une espèce hérite de toutes les méthodes de ses parents. Les collections permettent de coder des domaines particuliers. Une collection est associée à une espèce complètement définie. Toutes ses méthodes doivent être définies et toutes ses obligations de preuve déchargées. Les entités sont les éléments manipulés à l'intérieur des espèces. Le type support (*carrier* en anglais) d'une espèce est le type de ses entités. Chaque espèce contient un unique type support, éventuellement hérité. Les méthodes représentent les opérations de base ou les propriétés des structures. Il y a quatre catégories de méthodes : le *type support* (déclaré sous forme d'un type abstrait, ou bien lié à un type de donné (**int**, **list (string)**,...)) – les *signatures* représentant les opérations *déclarées* (introduction du nom et du type de l'opération) – les *méthodes* représentant les opérations *définies* (introduction du nom, du type et de la définition de l'opération) – les *propriétés* (déclarées) et les *théorèmes* (prouvés, donc définis) vérifiés par l'espèce et impliquant des obligations de preuve. De plus, à toute espèce est implicitement associée une interface,

obtenue en effaçant le corps des méthodes définies (fonctions ou théorèmes).

7.2 Implantation du cadre générique

On commence par définir les sujets, les objets et les modes d'accès comme des **setoid**, c'est-à-dire principalement comme des espèces déclarant une méthode d'égalité **equal**, un élément **element** et le type support **rep**.

```
species subjects inherits setoid = end

species objects inherits setoid = end

species access_mode inherits setoid =

  sig read in self;

  sig write in self;

end
```

L'espèce **access_mode** déclare les deux modes d'accès **read** et **write** qui seront les seuls considérés par souci de simplicité et de compatibilité avec les autres modèles de contrôle d'accès.

On définit ensuite les accès qui regroupent les sujets, les objets et les modes d'accès.

```
species access(s is subjects, o is objects, m is access_mode)
  inherits setoid =

  sig get_s in self -> s;

  sig get_o in self -> o;

  sig get_m in self -> m;

  sig create in s -> o -> m -> self;

  property access_equal :
    all s1 s2 in s, all o1 o2 in o, all m1 m2 in m,
      !equal(!create(s1, o1, m1), !create(s2, o2, m2)) ->
        (s!equal(s1, s2) and o!equal(o1, o2) and
         m!equal(m1, m2));

end
```

La propriété **access_equal** énonce simplement que si deux accès sont égaux alors leurs sujets, objets et modes d'accès respectifs sont égaux. Notons que le caractère **!** précède toujours l'appel à une méthode déclarée ou définie au sein d'une espèce.

On introduit ensuite les décisions définies de la même manière que les modes d'accès.

```
species decisions inherits setoid =

  sig yes in self;

  sig no in self;

  property yes_is_not_no :
    all x in self,
      not (!equal(x, !yes) and !equal(x, !no));

end
```

La propriété **yes_is_not_no** décrit le fait que les décisions **yes** et **no** sont différentes. Cette propriété est nécessaire pour exprimer cette différence car il n'est pas possible de déclarer un type inductif au sein d'une espèce.

Puis, les requêtes générales définissant la notion de requête sont introduites.

```
species requests_gal(s is subjects, o is objects,
                     m is access_mode)
  inherits requests =

  sig get_s in self -> s;

  sig get_o in self -> o;

  sig get_m in self -> m;

  sig is_get in self -> bool;

  sig is_rel in self -> bool;

  property get_or_rel :
    all x in self,
      !is_get(x) or !is_rel(x);

  property get_is_not_rel :
    all x in self,
      not (!is_get(x) and !is_rel(x));

end
```

Cette espèce hérite de l'espèce **requests** qui ne fait elle-même que d'hériter de **setoid**. Elle déclare les accesseurs des différentes composantes d'une requête et un prédicat **is_get** (resp. **is_rel**) renvoyant **true** si la requête prise en argument est une requête d'ajout (resp. de relâchement) d'accès. Comme précédemment, les propriétés **get_or_rel** et **get_is_not_rel** sont définies afin d'exprimer que soit une requête ajoute un accès, soit elle relâche un accès.

On définit ensuite la sémantique des requêtes associée aux requêtes générales.

```
species
  semantics_requests_gal(rh is rho, s is subjects, o is objects,
                         m is access_mode, a is access(s,o,m),
                         s_a is set_of(a),
                         r is requests_gal(s,o,m))
  inherits semantics_requests(rh,s,o,m,a,s_a,r) =

  letprop sem_req_get(r1, st1) =
    r!is_get(r1) ->
      s_a!est_element(a!create(r!get_s(r1), r!get_o(r1),
                               r!get_m(r1)),
                      !lambda(st1));

  letprop sem_req_rel(r1, st1) =
    r!is_rel(r1) ->
      not s_a!est_element(a!create(r!get_s(r1), r!get_o(r1),
                                   r!get_m(r1)),
                          !lambda(st1));

  letprop sem_req(r1, st1) =
    !sem_req_get(r1, st1) and !sem_req_rel(r1, st1);

end
```

L'espèce **rho** représente un paramètre de sécurité générique qui réalise uniquement un héritage de **setoid**. De plus, l'espèce paramétrée **set_of(a)** décrit un ensemble fini d'accès. C'est une espèce intermédiaire qui fait le lien avec la bibliothèque d'ensembles finis développée par Mathieu Jaume et Virgile Prevosto. Quant à l'espèce paramétrée **semantics_requests**, c'est aussi une espèce générique qui déclare dans son corps la propriété paramétrée **sem_req** qui représente la sémantique des requêtes. Comme nous pouvons le voir, **sem_req** est la conjonction des propriétés paramétrées **sem_req_get** et **sem_req_rel** qui traduisent exactement la sémantique des requêtes définie en (5).

7.3 Implantation du modèle RBAC96

Nous présentons maintenant la partie de l'implantation qui correspond à la formalisation de RBAC96.

Ainsi, nous introduisons les espèces représentant les utilisateurs, les rôles et le paramètre de sécurité ρ_{rbac}.

```
species users inherits setoid = end

species roles inherits partial_order = end

species rho_rbac(u is users, ro is roles) inherits rho = end
```

Comme l'espèce **subjects**, l'espèce **users** hérite de **setoid**. Par contre, l'espèce **roles** hérite de l'espèce **partial_order**, définie dans la bibliothèque

du langage, car l'ensemble des rôles est muni d'une relation d'ordre partiel. Enfin, l'espèce paramétrée **rho_rbac** est une traduction directe du paramètre ρ_{rbac}.

L'espèce suivante représente les états appartenant à Σ_{rbac}.

```
species states_rbac(u is users, ro is roles,
                    rh is rho_rbac(u,ro),
                    s is subjects, o is objects,
                    m is access_mode, a is access(s,o,m),
                    s_a is set_of(a))
  inherits states(rh,s,o,m,a,s_a) =

  sig user in self -> s -> u;

  sig ua in self -> u -> ro -> bool;

  sig pa in self -> o -> m -> ro -> bool;

  sig roles_s in self -> s -> ro -> bool;

  sig exists_role in self -> (ro -> bool) ->
                             (ro -> bool) -> bool;

  property add_access_ua :
    all st1 st2 in self, all s1 s2 in s, all o1 in o,
     all m1 in m, all ro1 in ro,
      !equal(st2, !add(st1, a!create(s1, o1, m1))) ->
         !ua(st1, !user(st1, s2), ro1) ->
           !ua(st2, !user(st2, s2), ro1);

  property del_access_ua :
    all st1 st2 in self, all s1 s2 in s, all o1 in o,
     all m1 in m, all ro1 in ro,
      !equal(st2, !del(st1, a!create(s1, o1, m1))) ->
         !ua(st1, !user(st1, s2), ro1) ->
           !ua(st2, !user(st2, s2), ro1);

  ...

end
```

Ainsi, l'espèce **states_rbac** hérite de l'espèce **states** qui déclare la fonction **lambda** renvoyant les accès courants d'un état et déclare les fonctions de sécurité d'un état. Notons que chaque fonction de sécurité a pour premier argument un état (représenté par **self**), étant donné que le calcul qu'elle effectue correspond à un état donné et peut donc évoluer. La fonction **user** correspond exactement à la fonction *user* de la formalisation. Quant aux fonctions **ua**, **pa** et **roles_s**, elles représentent respectivement les fonctions caractéristiques des ensembles calculés par les relations UA, PA et *roles*. L'utilisation de fonctions caractéristiques est un choix d'implantation destiné à une simplification du code et des preuves en évitant de manipuler explici-

tement des représentations d'ensembles. De plus, le prédicat **exists_role** , qui nous sera utile dans la définition de la fonction de transition, est déclaré. Il a pour spécification d'être vérifié si et seulement si il existe un rôle satisfaisant deux prédicats pris en paramètre. Enfin, pour chaque fonction de sécurité, deux propriétés sont définies stipulant que le résultat calculé par une fonction de sécurité ne dépend pas des accès courants.

Nous définissons à présent l'espèce implantant la politique $\mathbb{P}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}]$.

```
species policy_rbac(u is users, ro is roles,
                    rh is rho_rbac(u,ro),
                    s is subjects, o is objects,
                    m is access_mode, a is access(s,o,m),
                    s_a is set_of(a))
  inherits
    states_rbac(u,ro,rh,s,o,m,a,s_a),
    policy(rh,s,o,m,a,s_a) =

  letprop er(st in self, sp in s, r1 in ro) =
    ex r2 in ro,
      ro!leq(r1, r2) and !ua(st, !user(st, sp), r2);

  letprop ep(st in self, sp in s, op in o, mo in m) =
    ex r1, r2 in ro,
      !roles_s(st, sp, r1) and ro!leq(r2, r1) and
      !pa(st, op, mo, r2);

  letprop roles_er(st in self) =
    all sp in s, all r in ro,
      !roles_s(st, sp, r) ->
        !er(st, sp, r);

  letprop perm_ep(st in self) =
    all sp in s, all op in o, all mo in m,
      s_a!est_element(a!create(sp, op, mo), !lambda(st)) ->
        !ep(st, sp, op, mo);

  letprop omega(st in self) =
    !roles_er(st) and !perm_ep(st);

  ...

end
```

Les propriétés paramétrées **er** et **ep** traduisent les fonctions ER et EP avec une vision fonction caractéristique étant donné qu'elles reposent sur les fonctions de sécurité définies dans l'espèce **states_rbac**. De même, les propriétés paramétrées **roles_er** et **perm_ep** représentent respectivement les propriétés (1) et (2). Enfin, la propriété paramétrée **omega** représente bien le prédicat de sécurité Ω_{rbac} étant donné qu'elle est la conjonction de **roles_er** et **perm_ep**.

Enfin, nous définissons l'espèce représentant le modèle $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}]$.

```
species models_rbac(u is users, ro is roles,
                    rh is rho_rbac(u,ro),
                    s is subjects, o is objects,
                    m is access_mode, a is access(s,o,m),
                    s_a is set_of(a), r is requests_gal(s,o,m),
                    d is decisions)
  inherits
    policy_rbac(u,ro,rh,s,o,m,a,s_a),
    semantics_requests_gal(rh,s,o,m,a,s_a,r),
    models(rh,s,o,m,a,s_a,r,d) =

  let tau_rbac_acc(r1 in r, st1 in self) =
     let s1 = r!get_s(r1) in
     let o1 = r!get_o(r1) in
     let m1 = r!get_m(r1) in
     if r!is_get(r1) then
       if !exists_role(st1, !roles_s(st1, s1),
                            !pa(st1, o1, m1)) then
         (d!yes, !add(st1, a!create(s1, o1, m1)))
       else (d!no, st1)
     else (d!yes, !del(st1, a!create(s1, o1, m1)));

  theorem tau_rbac_acc_secure :
    all st1 st2 in self, all r1 in r, all d1 in d,
      !tau_rbac_acc(r1, st1) = (d1, st2) ->
        !omega(st1) ->
          !omega(st2)
    proof:
      <1>1 assume st1 st2 in self
                  r1 in r
                  d1 in d
                  H1: !tau_rbac_acc(r1, st1) = (d1, st2)
                  H2: !omega(st1)
           prove !omega(st2)
              ...
      <1>f qed;

  theorem tau_rbac_acc_r_correct :
    all st1 st2 in self, all r1 in r, all d1 in d,
      !tau_rbac_acc(r1, st1) = (d1, st2) ->
        d!equal(d1, d!yes) ->
          !sem_req(r1, st2)
    proof:
      <1>1 assume st1 st2 in self
                  r1 in r
                  d1 in d
                  H1: !tau_rbac_acc(r1, st1) = (d1, st2)
           prove d!equal(d1, d!yes) ->
                 !sem_req(r1, st2)
              ...
      <1>f qed;
```

```
  theorem tau_rbac_acc_r_correct_bis :
    all st1 st2 in self, all r1 in r, all d1 in d,
      !tau_rbac_acc(r1, st1) = (d1, st2) ->
        d!equal(d1, d!no) ->
          !equal(st1, st2)
    proof:
      <1>1 assume st1 st2 in self
                  r1 in r
                  d1 in d
                  H1: !tau_rbac_acc(r1, st1) = (d1, st2)
           prove d!equal(d1, d!no) ->
                 !equal(st1, st2)
                ...
      <1>f qed;

  ...

end
```

La fonction **tau_rbac_acc** implante la fonction de transition $\tau^{acc}_{\mathsf{rbac}}$ avec cependant une restriction. En effet, le test contrôlant l'ajout d'un accès repose sur la fonction **exists_role** qui vérifie qu'il existe un rôle satisfaisant les prédicats **roles_s** et **pa**. Or, ce calcul est une simplification du calcul d'ensemble effectué par la fonction EP car il ne tient pas compte de la hiérarchie des rôles. Néanmoins, en implantant la fonction **pa** de manière à ce qu'elle reconnaisse tous les couples permission/rôle autorisés (c'est-à-dire en codant « à la main » l'héritage de permissions réalisé par la hiérarchie des rôles), nous obtenons une fonction **exists_role** qui caractérise un ensemble identique à celui défini par EP, et donc dans ce cas, la fonction **tau_rbac_acc** respecte exactement la formalisation. De plus, trois théorèmes sont définis. Le premier énonce que la fonction de transition est correcte vis-à-vis du prédicat de sécurité et les deux suivants qu'elle respecte la sémantique des requêtes. Les preuves, qui ont constituées la difficulté majeure du développement, n'ont pas été reportées ici en raison de leur longueur.

7.4 Implantation de l'exemple

Nous présentons enfin la dernière partie de l'implantation qui concerne la concrétisation des espèces exposées plus haut permettant d'aboutir au codage en Focal de l'exemple de la section précédente.

Ainsi, nous commençons par introduire l'espèce **s_setoid** qui est utilisée pour concrétiser plusieurs espèces comme nous le verrons par la suite. Cette espèce nous permet donc d'éviter de dupliquer du code.

```
species s_setoid inherits setoid =

  rep = int * string;

  let create(n in int, s in string) in self = (n, s);

  let equal(s1, s2) = #first(s1) = #first(s2);

  let element = (0, "element");

  let print(s) = #scnd(s);

  proof of equal_reflexive = by #beq_refl def !equal;

  proof of equal_symmetric = by #beq_symm def !equal;

  proof of equal_transitive = by #beq_trans def !equal;

end
```

Elle définit le type **rep** comme un couple entier/chaîne de caractères ainsi que les méthodes nécéssaires à la concrétisation d'un **setoid** (ce qui inclus les preuves des propriétés de la fonction **equal**). L'entier est utilisé pour différencier clairement les entités (par exemple, pour réaliser des preuves) tandis que la chaîne de caractère servira à la fonction d'impression. De plus, une fonction **create**, qui est un constructeur du type **self**, est définie. Notons que le caractère # précède toujours l'appel à une méthode définie hors d'une espèce (au toplevel).

Une première utilisation de l'espèce **s_setoid** est illustrée par les trois espèces suivantes qui concrétisent les notions de sujets, d'objets et d'utilisateurs en héritant uniquement de leurs espèces génériques respectives et de **s_setoid**.

```
species s_subjects inherits subjects, s_setoid = end

species s_objects inherits objects, s_setoid = end

species s_users inherits users, s_setoid = end
```

De manière similaire, nous définissons les trois espèces suivantes.

```
species s_access_mode inherits access_mode, s_setoid =

  let read = (1, "read");

  let write = (2, "write");

end
```

```
species s_decisions inherits decisions, s_setoid =

  let yes = (1, "yes");

  let no = (2, "no");

  proof of yes_is_not_no = assumed;

end

species s_get_release inherits s_setoid =

  let get in self = (1, "+");

  let rel in self = (2, "-");

end
```

Ces espèces fixent les entités qui doivent les composer contrairement aux trois espèces précédentes dont les entités seront définies par l'utilisateur grâce à la fonction **create**. La preuve de la propriété **yes_is_not_no** n'a pas été réalisée étant donné que Zenon, le prouveur de l'atelier Focal, ne permet pas d'effectuer des preuves sur les entiers.

L'espèce suivante concrétise la notion d'accès en raffinant l'espèce générique **access**.

```
species s_access(s_sub is s_subjects,
                 s_obj is s_objects,
                 s_acc_mod is s_access_mode)
  inherits access(s_sub,s_obj,s_acc_mod) =

  rep = (s_sub * (s_obj * s_acc_mod));

  let create(s, o, m) = (s, (o, m));

  let get_s(a) = #first(a);

  let get_o(a) = #first(#scnd(a));

  let get_m(a) = #scnd(#scnd(a));

  let equal(a1, a2) =
    #and_b(#and_b(s_sub!equal(!get_s(a1), !get_s(a2)),
                  s_obj!equal(!get_o(a1), !get_o(a2))),
           s_acc_mod!equal(!get_m(a1), !get_m(a2)));

  let element = !create(s_sub!element,
                        s_obj!element,
                        s_acc_mod!element);
```

```
  let print(a) =
    "(" ^ s_sub!print(!get_s(a)) ^
    "," ^ s_obj!print(!get_o(a)) ^
    "," ^ s_acc_mod!print(!get_m(a)) ^
    ")";

  ...

end
```

Un accès est représenté comme un triplet sujet/objet/mode d'accès comme cela a été défini dans la formalisation.

De même, nous concrétisons l'implantation d'une requête sur les accès (i.e. appartenant à $\mathcal{R}^{acc}$).

```
species s_requests_gal(s is s_subjects,
                       o is s_objects,
                       m is s_access_mode,
                       sgr is s_get_release)
  inherits requests_gal(s,o,m) =

  rep = (sgr * (s * (o * m)));

  let create(g, sa, oa, ma) in self = (g, (sa, (oa, ma)));

  let get_gr(rg in self) in sgr = #first(rg);

  let get_s(rg) = #first(#scnd(rg));

  let get_o(rg) = #first(#scnd(#scnd(rg)));

  let get_m(rg) = #scnd(#scnd(#scnd(rg)));

  let is_get(rg) =
    sgr!equal(sgr!get, !get_gr(rg));

  let is_rel(rg) =
    sgr!equal(sgr!rel, !get_gr(rg));

  let equal(rg1, rg2) = ...;

  let element = ...;

  let print(rg) = ...;

  ...

end
```

Nous définissons ensuite la concrétisation de la notion de rôles.

```
species s_roles inherits roles, s_setoid =

  let directeur in self = (1, "directeur");

  let secretaire in self = (2, "secretaire");

  ...

  let roles_list in list(self) =
     [
      !directeur; !secretaire;
      !chirurgienChef; !chirurgien;
      !radiologiste; !radiologisteAssistant;
      !specialiste; !generaliste;
      !infirmiere; !secretaireMedicale;
      !patient
     ];

  let leq(r1, r2) =
     if !equal(r1, !directeur) then
       if !equal(r2, !directeur) then #True
       else #False
     ...;

  ...

end
```

Les rôles introduits dans l'exemple sont directement définis au sein de l'espèce car nous devons pouvoir fixer l'ensemble des rôles considérés (donné par la liste **roles_list**) et l'ordre partiel associé à cet ensemble (modélisé par la fonction **leq**).

Nous pouvons à présent introduire la concrétisation du paramètre de sécurité ρ_{rbac}.

```
species s_rho_rbac(s_us is s_users, s_ro is s_roles)
  inherits rho_rbac(s_us,s_ro), s_setoid = end
```

Enfin, nous définissons la concrétisation du modèle $\mathbb{M}_{\mathsf{rbac}}[\rho_{\mathsf{rbac}}]$.

```
species s_models_rbac(s_us is s_users,
                      s_ro is s_roles,
                      s_rh is s_rho_rbac(s_us,s_ro),
                      s_sub is s_subjects,
                      s_obj is s_objects,
                      s_acc_mod is s_access_mode,
                      s_acc is s_access(s_sub,s_obj,s_acc_mod),
                      s_set_acc is s_set_of(s_acc),
                      s_gr is s_get_release,
                      s_req is s_requests_gal(s_sub,s_obj,
                                              s_acc_mod,s_gr),
                      s_dec is s_decisions)
```

```
inherits models_rbac(s_us,s_ro,s_rh,s_sub,s_obj,s_acc_mod,
                     s_acc,s_set_acc,s_req,s_dec) =

rep = (s_set_acc * ((s_sub -> s_us) *
                    (list(s_us * s_ro) *
                     (list((s_obj * s_acc_mod) * s_ro) *
                      list(s_sub * s_ro)))));

let create(s, ur, u, p, r) in self = (s, (ur, (u, (p, r))));

let lambda(st) = #first(st);

let upsilon(st in self) = #scnd(st);

let add(st, ac) =
   (s_set_acc!union(!lambda(st), s_set_acc!singleton(ac)),
    !upsilon(st));

let del(st, ac) =
   (s_set_acc!diff(!lambda(st), s_set_acc!singleton(ac)),
    !upsilon(st));

let user(st, sub) = #first(!upsilon(st))(sub);

let ua(st, us, ro) =
   let rec aux(l) =
      match l with
      | [] -> #False
      | h::t ->
          if #and_b(s_us!equal(us, #first(h)),
                    s_ro!equal(ro, #scnd(h))) then #True
          else aux(t)
      end
   in aux(#first(#scnd(!upsilon(st))));

let pa(st, obj, mod, ro) = ...;

let roles_s(st, sub, ro) = ...;

let exists_role(st, fr, fp) =
   let rec aux(l) =
      match l with
      | [] -> #False
      | h::t ->
          if #and_b(fr(h), fp(h)) then #True
          else aux(t)
      end
   in aux(s_ro!roles_list);

let equal(st1, st2) =
   s_set_acc!equal(!lambda(st1), !lambda(st2));
```

```
  let element =
     !create(s_set_acc!vide, (fun su -> s_us!element),
              [], [], []);

  let print(st) = s_set_acc!print(!lambda(st));

  ...

end
```

Cette concrétisation respecte exactement la formalisation étant donné que nous représentons un modèle par un état. En effet, le type **rep** est bien un 5-uplet comprenant un ensemble d'accès, une fonction associant un utilisateur à un sujet (pour *user*), une liste de couples utilisateur/rôle (pour UA), une liste de couples permission/rôle (pour PA) et une liste de couples sujet/rôle (pour *roles*). De plus, la fonction **lambda** renvoi bien l'ensemble d'accès de l'état, tandis que la fonction **upsilon** retourne ses fonctions de sécurité. Les définitions des fonctions **pa** et **roles_s** ont été omises étant donné qu'elles sont similaires à celle de **ua** où nous voyons que la fonction récursive interne **aux**, qui réalise le calcul, prend bien en paramètre la liste correspondant à UA. Quant à la fonction **exists_role** , elle peut trouver un rôle satisfaisant deux prédicats grâce au parcourt de l'ensemble des rôles donné par la liste **roles_list** .

Enfin, nous définissons l'ensemble des collections qui encapsulent les espèces décrites précédemment.

```
collection c_subjects implements s_subjects = end

collection c_objects implements s_objects = end

collection c_access_mode implements s_access_mode = end

collection c_access
  implements s_access(c_subjects,c_objects,c_access_mode) = end

collection c_set_of_access implements s_set_of(c_access) = end

collection c_decisions implements s_decisions = end

collection c_get_release implements s_get_release = end

collection c_requests_gal
  implements s_requests_gal(c_subjects,c_objects,c_access_mode,
                            c_get_release) = end

collection c_users implements s_users = end

collection c_roles implements s_roles = end
```

```
collection c_rho_rbac
  implements s_rho_rbac(c_users, c_roles) = end

collection c_models_rbac
  implements s_models_rbac(c_users, c_roles, c_rho_rbac,
                           c_subjects, c_objects, c_access_mode,
                           c_access, c_set_of_access,
                           c_get_release, c_requests_gal,
                           c_decisions) = end
```

Nous montrons maintenant comment nous utilisons ces collections pour terminer l'implantation de l'exemple.

```
let s1 = c_subjects!create(1, "s1");;

...

let dossierMedecin_1 = c_objects!create(1, "dossierMedecin_1");;

...

let alice = c_users!create(1, "alice");;

...
```

Nous créons de même les sujets, objets et utilisateurs restants.

Afin de créer un modèle de contrôle d'accès (représenté par un état), nous commençons par définir les fonctions de sécurité de l'exemple.

```
let user_ex(s in c_subjects) in c_users =
   if c_subjects!equal(s, #s1) then #alice
   else if c_subjects!equal(s, #s2) then #bob
   else if c_subjects!equal(s, #s3) then #charly
   else if c_subjects!equal(s, #s4) then #dalia
   else if c_subjects!equal(s, #s5) then #dalia
   else if c_subjects!equal(s, #s6) then #estel
   else if c_subjects!equal(s, #s7) then #franck
   else if c_subjects!equal(s, #s8) then #gregory
   else if c_subjects!equal(s, #s9) then #helen
   else if c_subjects!equal(s, #s10) then #isabel
   else if c_subjects!equal(s, #s11) then #john
   else if c_subjects!equal(s, #s12) then #john
   else c_users!element;;
```

```
let ua_ex in list(c_users * c_roles) =
   [
    (#alice, c_roles!directeur);
    (#bob, c_roles!chirurgienChef);
    (#charly, c_roles!radiologiste);
    (#dalia, c_roles!chirurgien);
    (#estel, c_roles!secretaireMedicale);
    (#franck, c_roles!chirurgien);
    (#gregory, c_roles!generaliste);
    (#helen, c_roles!secretaire);
    (#isabel, c_roles!infirmiere);
    (#john, c_roles!radiologisteAssistant)
   ];;

let pa_ex in list((c_objects * c_access_mode) * c_roles) = ...;;

let roles_s_ex in list(c_subjects * c_roles) = ...;;
```

Nous voyons que la correspondance avec l'exemple est immédiate.

Enfin, nous exposons une partie du code de l'exemple d'exécution qui correspond à la première transition de la figure 2.

```
#print_string("Exemple d'execution :\n\n");;

let st0 = c_models_rbac!create(c_set_of_access!vide,
                               #user_ex,
                               #ua_ex,
                               #pa_ex,
                               #roles_s_ex);;

#print_string("Acces0 : " ^
              c_models_rbac!print(#st0) ^
              "\n");;

let req0 = c_requests_gal!create(c_get_release!get,
                                 #s1,
                                 #dossierMedecin_1,
                                 c_access_mode!write);;

#print_string("Requete0 : " ^
              c_requests_gal!print(#req0) ^
              "\n");;

let rep0 = c_models_rbac!tau_rbac_acc(#req0, #st0);;

let dec0 = #first(#rep0);;

let st1 = #scnd(#rep0);;

#print_string("Reponse0 : Decision0 : " ^
              c_decisions!print(#dec0) ^
              "\n");;
```

```
#print_string("              Acces1 : " ^
              c_models_rbac!print(#st1) ^
              "\n");;
```

Son exécution produit le résultat suivant.

```
Exemple d'execution :

Acces0 : []
Requete0 : <+,s1,dossierMedecin_1,write>
Reponse0 : Decision0 : yes
           Acces1 : [(s1,dossierMedecin_1,write)]
```

8 Conclusion

La sécurité, et plus particulièrement le contrôle d'accès, sont des problématiques actuelles en informatique. En effet, il devient aujourd'hui important de pouvoir contrôler les flots d'informations dans les réseaux et dans les systèmes d'information. Il convient de développer au sein des systèmes informatiques des mécanismes permettant de filtrer les accès afin de ne laisser passer que ceux autorisés. Il s'agit pour cela de définir une politique de sécurité, c'est-à-dire la caractérisation des accès permis. Le programme chargé de mettre en application cette politique, le moniteur de référence, est souvent considéré comme l'une des clés de voûte de la sécurité d'un système. Sa conception et son développement doivent être menés de manière à garantir sa fiabilité et sa sûreté. En effet, toute faille au sein de ce programme pourrait entraîner des violations de la politique de sécurité.

Nous avons réalisé un travail comprenant trois axes principaux. Le premier a consisté à formaliser le modèle de contrôle d'accès RBAC96 en utilisant un cadre sémantique générique préexistant déjà utilisé pour étudier formellement d'autres modèles de contrôle d'accès. Dans un second temps, nous avons utilisé notre formalisation du modèle RBAC96 afin de le comparer avec le modèle de Bell et LaPadula en nous appuyant sur les outils de comparaison fournis par le cadre. Enfin, nous avons validé ce travail de formalisation en réalisant une implantation de notre modélisation et d'un exemple concret dans l'atelier Focal.

La formalisation du modèle RBAC96 a abouti à la définition de deux fonctions de transition principales. L'une, classique, concerne les requêtes permettant d'accèder aux objets d'un système tandis que l'autre, permet de prendre en compte des requêtes administratives ce qui représente un moyen intéressant d'administrer un système mettant en œuvre une politique à base de rôles. La prise en compte des requêtes administratives introduite dans ce travail ne figure pas dans les descriptions classiques du modèle RBAC96. En outre, nous avons montré formellement que les implantations basées sur ces fonctions de transition sont correctes vis-à-vis du modèle RBAC96.

La comparaison des modèles de RBAC96 et de Bell et LaPadula nous a permis de montrer deux résultats principaux. Tout d'abord, que le modèle de Bell et LaPadula est plus restrictif que celui de RBAC96 et ensuite, que le modèle de RBAC96 n'est pas plus restrictif que celui de Bell et LaPadula. Ainsi, nous avons montré que le modèle de Bell et LaPadula est moins expressif que celui de RBAC96, c'est-à-dire qu'il existe des états d'un système régit par la politique RBAC96 ne pouvant être représentés dans un système contrôlé par la politique de Bell et LaPadula. Ces deux résultats apportent un nouveau point de vue dans la compréhension des deux modèles comparés. D'autre part, afin de réaliser cette comparaison, nous avons fourni une interprétation à base de rôles du modèle de Bell et LaPadula qui peut s'avérer utile dans un contexte de réutilisation de code. En effet, cette interprétation a montré qu'il est possible d'exprimer le modèle de Bell et LaPadula avec les concepts de RBAC96, et donc de réutiliser une implantation d'un modèle à base de rôles pour coder un modèle à base de treillis (comme celui de Bell et LaPadula).

Parallèlement à la formalisation, nous avons réalisé une implantation de notre travail dans l'atelier Focal. Cette implantation illustre notamment deux possibilités offertes par Focal. La première consiste à effectuer un développement par raffinement qui privilégie la réutilisation. En effet, l'implantation comporte trois couches. Tout d'abord, une première architecture d'espèces représente le cadre sémantique générique, puis un raffinement est opéré afin d'obtenir une implantation du modèle RBAC96 et enfin, ces deux couches sont concrétisées par une troisième qui implante l'exemple de la section 6. Les espèces caractérisant cet exemple sont elles-même facilement réutilisables étant donné que pour implanter un exemple différent, il suffit de redéfinir uniquement l'espèce spécifiant les rôles du système. La deuxième possibilité exploitée, et non la moindre, concerne la preuve de théorèmes portant sur les aspects essentiels de l'implantation qui nous permet d'avoir une plus grande confiance dans notre développement.