Understanding and Developing Role-Based Administrative Models

Jason Crampton

Technical Report
RHUL–MA–2005–06
20 April 2005

Royal Holloway
University of London

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England
`http://www.rhul.ac.uk/mathematics/techreports`

Abstract

Access control data structures generally need to evolve over time in order to reflect changes to security policy and personnel. An administrative model defines the rules that control the state changes to an access control model and the data structures it defines. We present a powerful framework for describing role-based administrative models. The framework is based on the concept of administrative domains and state changes that preserve certain aspects of those domains. We define a number of different sets of criteria, each of which control the effect of state changes on the set of administrative domains and thereby lead to different role-based administrative models. Using this framework we are able to identify some unexpected connections between the ARBAC97 and RHA administrative models and to compare their respective properties. We also suggest some improvements to both models.

1 Introduction

An *access control mechanism* is a component of a computer system that is used to limit the access that authenticated and authorized users have to the resources available to users of the computer system. An *access control model* typically defines a collection of sets, functions and relations that represent elements of an access control mechanism. We will refer to such sets as *components* of the access control model. The components of the Harrison-Ruzzo-Ullman (HRU) model [6], for example, are the set of objects O, the set of subjects S, the set of access rights A, and the protection matrix $M : S \times O \to 2^A$.

A component can be *static* or *dynamic*: a static component is one that doesn't change over time, such as the set of access rights A in the protection matrix model; conversely, a dynamic component, such as the set of subjects S in the protection matrix model, does change over time. The *state* of a model can be thought of as a tuple $(C_1, \ldots, C_n)$, where C_j is a dynamic component in the model. In the HRU model, for example, the state is defined to be the tuple (S, O, M); the set of access rights is static.

An administrative model for an access control mechanism defines a decision process that determines whether a request to change the state is permitted. Typically, an administrative model is defined by a fixed set of commands, each command containing a conditional statement and a body that is executed if the conditional statement evaluates to true. The body will comprise a number of atomic operations each of which changes a dynamic component of the associated access control model. In the HRU model, for example, the conditional statement checks for the presence of access rights in the matrix and the atomic operations make changes to the rows, columns or entries of the matrix.

Role-based access control (RBAC) models have been the subject of considerable research in recent years resulting in several important models, including the RBAC96 model [12], the role graph model [9], and the NIST model [5]. Many of these ideas were recently consolidated to form the basis for the ANSI RBAC standard [1].

Despite the enthusiasm for RBAC, the use of RBAC principles to manage RBAC systems has been less widely studied. The models cited above, for example, rely on centralized procedures to change dynamic components of the respective models. The most mature decentralized role-based administrative models are ARBAC97 [11] and the RHA (role hierarchy administration) model [4]. Both models are designed to inter-operate with RBAC96, but could also be applied to ANSI RBAC systems, and both models exploit the structure of the role hierarchy to control changes. Nevertheless, the motivation for the development and design of these models has been somewhat vague: the creators of ARBAC97, for example, talk of the need to prevent "anomalous side effects" arising from unconstrained changes to the hierarchy; similarly, Crampton and Loizou state that RHA prevents "unexpected side effects due to inheritance elsewhere in the hierarchy", although this notion is formalized to some extent.

In this paper, we undertake a rigorous analysis of the properties that define the behaviour of role-based administrative models. The analysis is based on the notion of an administrative domain, a self-contained sub-hierarchy of the role hierarchy. Our first major result is to prove that administrative domains are pairwise nested or disjoint. We then define what it means for administrative domains to be preserved. This enables us to define a number of sets of criteria that impose constraints on the functionality of a role-based administrative model. Each of these sets defines a mode of operation for a role-based administrative model, enabling systems and application developers to choose the administrative model best suited to their requirements. Each set of criteria requires that administrative operations must preserve certain structural properties of administrative domains. We might insist, for example, that an administrative role can make changes to its own administrative domain and any domains contained within it. These criteria enable us to classify role-based administrative models according to their permissiveness, and to create a framework for developing role-based administrative models.

One of the most striking consequences of our analysis is to reveal a fundamental and hitherto unexpected connection between RHA and ARBAC97. In particular, we find that RHA is the most permissive of administrative models, whereas ARBAC97 is among the most restrictive. Informally, our analysis enables us to draw a road map from RHA to ARBAC97, identifying interesting features (new models) along the way. We also establish that ARBAC97 is more restrictive than is necessary and point out a number of weaknesses in the original formulation.

In the next section we briefly review the RBAC96 model and some relevant mathematics. We also identify and specify the operational semantics of the primitive operations used in a role-based administrative model. In Section 3 we define administrative scope, the central concept in RHA, and show how this immediately leads to the concept of an administrative domain. We prove that administrative domains must be either disjoint or nested and introduce the idea of a domain tree. In Section 4 we formally define what it means for an administrative domain to be preserved by an administrative operation and introduce the idea of local, hierarchical and universal preservation. We also define what it means for an operation to be autonomy preserving. We then introduce three different sets of criteria and state a number of important results in relation to the preservation of administrative domains. In Section 5 we introduce the idea of an administrative role and define a template for constructing role-based administrative models. In this section we note the connection between this template and RHA. In Section 6 we describe the connection between our framework and ARBAC97, and provide a more concise characterization of ARBAC97. We also identify a number of flaws in ARBAC97 and describe appropriate remedies. We conclude with an appraisal of our framework and describe the numerous opportunities for further research in this area.

2 Preliminaries

2.1 RBAC96

RBAC96 is a family of access control models that assumes the existence of a set of roles R, a set of permissions P, a set of roles U, and two relations $UA \subseteq U \times R$ and $PA \subseteq P \times R$ that bind users and permissions to roles [12]. These sets and relations form the basis for RBAC$_0$, the simplest model in the RBAC96 family. A request by a user u to invoke permission p is granted if there exists a role r such that $(u, r) \in UA$ and $(p, r) \in PA$.

RBAC$_1$ introduces the concept of a role hierarchy, which is modelled as a partial order on the set of roles. In other words, the role hierarchy is a binary relation $RH \subseteq R \times R$ that is reflexive, anti-symmetric and transitive. The role hierarchy permits a role r to inherit the permissions assigned to any role r' such that $r' < r$.[1] This significantly reduces the administrative burden by reducing the number of explicit assignments that need to be stored in the UA and PA relations. (RBAC$_0$ and RBAC$_1$ have recently been superseded by the core and hierarchical components of the ANSI RBAC standard [1].)

2.2 Partial orders

Let $\langle X, \leqslant \rangle$ be a partially ordered set and let $x, y \in X$. We write $x < y$ if $x \leqslant y$ and $x \neq y$. We may write $y \geqslant x$ whenever $x \leqslant y$. We write $x \parallel y$ if $x \nleqslant y$ and $y \nleqslant x$. We say Y is an *antichain* if for all $y, z \in Z$, $y \neq z$ implies that $y \parallel z$.

We say y *covers* x, or x is *covered by* y, denoted $x \lessdot y$, if $x < y$ and for all $z \in X$, $x \leqslant z < y$ implies $x = z$. In other words, $x \lessdot y$ is shorthand for "y is the immediate parent of x". The *Hasse diagram of* X is the directed graph $(X, \lessdot)$: in other words, transitive relationships in the poset are implied by paths in the Hasse diagram. In the context of RBAC, the Hasse diagram represents the role hierarchy.

We define $\triangle x = \{y \in X : y \lessdot x\}$ and $\triangledown x = \{y \in X : x \lessdot y\}$. In other words, $\triangle x$ is the set of immediate children of x and $\triangledown x$ is the set of immediate parents of x. It is easy to show that $\triangle x$ and $\triangledown x$ are antichains for all $x \in X$.

We define $\downarrow x = \{y \in X : y \leqslant x\}$ and $\uparrow x = \{y \in X : x \leqslant y\}$. For $Y \subseteq X$, we define

$$\downarrow Y = \bigcup_{y \in Y} \downarrow y \quad \text{and} \quad \uparrow Y = \bigcup_{y \in Y} \uparrow y.$$

[1]It is customary to write $r \leqslant r'$ if $(r, r') \in RH$ and $r < r'$ if $r \leqslant r'$ and $r \neq r'$.

In the context of RBAC, $\downarrow r$ represents the set of roles available to a user assigned to r and $\uparrow r$ represents the set of roles to which the permission p is available if p is assigned to r. The expression $\downarrow r \cup \uparrow r$ will be used extensively when we introduce administrative scope, and will be abbreviated to $\updownarrow r$.

2.3 Administrative operations

Role-based access control models typically include a role hierarchy, which is modelled as a partial order on the set of roles. The role hierarchy is represented as the set of directed edges in the Hasse diagram of R (an example is shown in Figure 1 on page 8). This gives rise to the following set of *hierarchy operations*:

- addEdge(a, c, p), which adds the directed edge (c, p) to the hierarchy, where $c, p \in R$;
- deleteEdge(a, c, p), which deletes the directed edge (c, p) from the hierarchy;
- addRole(a, r, C, P), which creates the role r with immediate children $C \subseteq R$ and immediate parents $P \subseteq R$;
- deleteRole(a, r), which deletes the role $r \in R$.

In addition we have the following *assignment operations*:

- addUA(a, u, r), which adds the pair (u, r) to the UA relation;
- deleteUA(a, u, r), which deletes the pair (u, r) from the UA relation;
- addPA(a, p, r), which adds the pair (p, r) to the PA relation;
- deletePA(a, p, r), which deletes the pair (p, r) from the PA relation.

Collectively we refer to these eight operations as *administrative operations*. In this paper, we will focus on the hierarchy operations; experience has shown that it is straightforward to incorporate the other operations [4].

Informally, the execution of a hierarchy operation will affect one or more roles in the hierarchy. The set of roles that are affected by an operation is not necessarily immediately obvious, because of the o, transitivity implied by the role hierarchy. It may be necessary, for example, to "repair" the hierarchy relation following addEdge and deleteEdge operations in order to remove redundancy and to preserve inheritance, respectively. Table 1 summarizes the changes to R and RH caused by hierarchy operations.

3 Administrative scope

The RBAC96 model does not provide any model for controlling updates to the role hierarchy and the assignment relations. This omission was addressed by the ARBAC97 model [11], which provides a role-based model for administering a role-based access control

Operation	**Semantics**
$\mathsf{addEdge}(a, c, p)$	$RH \leftarrow RH \cup \{(c,p)\} \setminus \{(x,p) : x \in \triangle c \cap \triangle p\} \setminus \{(c,y) : y \in \triangledown c \cap \triangledown p\}$ If $(r,c), (r,p) \in RH$ and the edge (c,p) is added, then we no longer require the edge (r,p), because (r,c) and (c,p) imply (r,p) by transitivity. Hence any role r that is an immediate child of both c and p is affected by the operation as the edge (r,p) must be deleted from RH. Similarly, any role r that is an immediate parent of both c and p is affected by $\mathsf{addEdge}(a,c,p)$ as the edge (c,r) must be deleted from RH.
$\mathsf{deleteEdge}(a, c, p)$	$RH \leftarrow RH \setminus \{(c,p)\} \cup \{(x,p) : x \in \triangle c\} \cup \{(c,y) : y \in \triangledown p\}$ If $(r,c) \in RH$, then $r < c < p$ and hence we may need to add the edge (r,p) to preserve the inheritance. Similarly, if $(p,r) \in RH$, then $c < p < r$ and we may need to add the edge (c,r).
$\mathsf{addRole}(a, r, C, P)$	$R \leftarrow R \cup \{r\}$ $RH \leftarrow RH \cup \{(c,r) : c \in C\} \cup \{(r,p) : p \in P\} \setminus \{(c,p) : c \in C, p \in P\}$ If $(c,p) \in RH$, where $c \in C$ and $p \in P$, then the edge (c,p) becomes redundant following the addition of role r (since (c,r) and (r,p) are added to the hierarchy, thereby implying $c < p$ by transitivity). Hence we remove (c,p) from RH.
$\mathsf{deleteRole}(a, r)$	$R \leftarrow R \setminus \{r\}$ $RH \leftarrow RH \cup \{(c,p) : c \in \triangle r, p \in \triangledown r\}$ For any role $c \in \triangle r$ and any role $p \in \triangledown r$ we have $c < r < p$, so we must add an edge (c,p) to the hierarchy following the deletion of r.

Table 1: Operational semantics of hierarchy operations

system. However, the ARBAC97 model suffers from its inability to manage many types of hierarchies [4, Section 8]. Crampton and Loizou introduced the RHA model[2] as a more flexible and widely applicable alternative to ARBAC97 [4].

The RHA model is based around the idea of *administrative scope*. Every role $r \in R$ has an administrative scope, denoted $\sigma(r)$, which defines the set of roles that can be modified by r. Administrative scope is determined by the structure of the hierarchy. Informally, $r' \in \sigma(r)$ if any change to r' will only be observed by r and roles more senior than r. That is, any change to r' made by r will not have unexpected side effects due to inheritance elsewhere in the hierarchy.

Definition 1 *The* administrative scope *of a role r is defined to be* [4]

$$\sigma(r) = \{s \in {\downarrow}r : {\uparrow}s \subseteq {\updownarrow}r\}.$$

[2]In fact there are four different RHA models of differing complexity. For convenience we will refer to the RHA model, except in Section 5, when we discuss the particular members of the family.

The strict administrative scope *of r is defined to be $\sigma(r) \setminus \{r\}$ and is denoted $\widehat{\sigma}(r)$. For $A \subseteq R$ we define $\sigma(A) = \{r \in {\downarrow}A : {\uparrow}r \subseteq {\updownarrow}A\}$ and $\widehat{\sigma}(A) = \sigma(A) \setminus A$.*

Note that $r \in \sigma(r)$ for all r, which motivates the definition of strict administrative scope. In the role hierarchy depicted in Figure 1 on page 8, for example, $\sigma(\texttt{PL1}) = \{\texttt{ENG1}, \texttt{PE1}, \texttt{QE1}, \texttt{PL1}\}$.

3.1 Administrative scope and administrative operations

The conditions that determine whether an administrative operation is allowed to proceed in the RHA model are summarized in Table 2.

Operation	**Conditions**
$\mathsf{addRole}(a, r, C, P)$	$C \subseteq \widehat{\sigma}(a), P \subseteq \sigma(a)$
$\mathsf{deleteRole}(a, r)$	$r \in \widehat{\sigma}(a)$
$\mathsf{addEdge}(a, c, p)$	$c, p \in \sigma(a)$
$\mathsf{deleteEdge}(a, c, p)$	$c, p \in \sigma(a)$

Table 2: Conditions for success of hierarchy operations in RHA

3.2 Administrative domains

We will say $D \subseteq R$ is an *administrative domain*, with *administrator* r, if $D = \sigma(r)$ for some $r \in R$. Since $r \in \sigma(r)$ for all $r \in R$, we say D is a *non-trivial* administrative domain if $|D| > 1$. Henceforth, we assume that all administrative domains are non-trivial. We will write $\mathcal{D}_R$ for the set of administrative domains in R. Henceforth we will omit R when it is obvious from context.

In this section we establish a fundamental result concerning administrative domains: namely, that domains are either nested or disjoint. This leads naturally to the concept of an administrative domain tree and of the smallest domain containing a given role. These concepts will be used extensively in the following section.

Lemma 2 *Let $a, b \in R$. Then*

$$\sigma(a) \cap \sigma(b) = \begin{cases} \sigma(a) & \text{if } a \in \sigma(b), \\ \sigma(b) & \text{if } b \in \sigma(a), \\ \emptyset & \text{otherwise.} \end{cases}$$

Proof Let $r \in \sigma(a)$. We consider each of the three cases in turn. Note that $\sigma(a) \cap \sigma(b) = \sigma(a)$ is equivalent to saying that $\sigma(a) \subseteq \sigma(b)$.

- Now if $a \in \sigma(b)$ then, by definition, $a \leqslant b$, $\downarrow a \subseteq \downarrow b$ and $\uparrow a \subseteq \updownarrow b$. Hence $\uparrow r \subseteq \updownarrow a \subseteq \updownarrow b$ and $r \in \sigma(b)$.
- By symmetry, $\sigma(a) \cap \sigma(b) = \sigma(b)$ if $b \in \sigma(a)$.
- Now assume that $a \notin \sigma(b)$ and $b \notin \sigma(a)$. There are three cases to consider:
  - If $a \leqslant b$, then since $a \notin \sigma(b)$, there exists $x \in \uparrow a$ such that $x \notin \updownarrow b$. Now $r \leqslant a$, by definition of $\sigma(a)$ and hence $r \leqslant x$ by transitivity. Therefore $x \in \uparrow r$ and hence $r \notin \sigma(b)$.
  - If $a \parallel b$, then $a \notin \updownarrow b$. By definition, $r \in \sigma(a)$ implies $r \leqslant a$ (that is, $a \in \uparrow r$) and hence $r \notin \sigma(b)$.
  - If $a > b$, suppose, in order to obtain a contradiction, that $r \in \sigma(b)$. Then, by definition, $r \leqslant b$ and $\uparrow r \supseteq \uparrow b$. Hence $\uparrow b \subseteq \uparrow r \subseteq \updownarrow a$, which implies that $b \in \sigma(a)$, the required contradiction.

■

Remark 3 *Note that $a \leqslant b$ does not imply that $\sigma(a) \subseteq \sigma(b)$. A counterexample is provided by* `ED` *and* `PL1` *in Figure 1.*

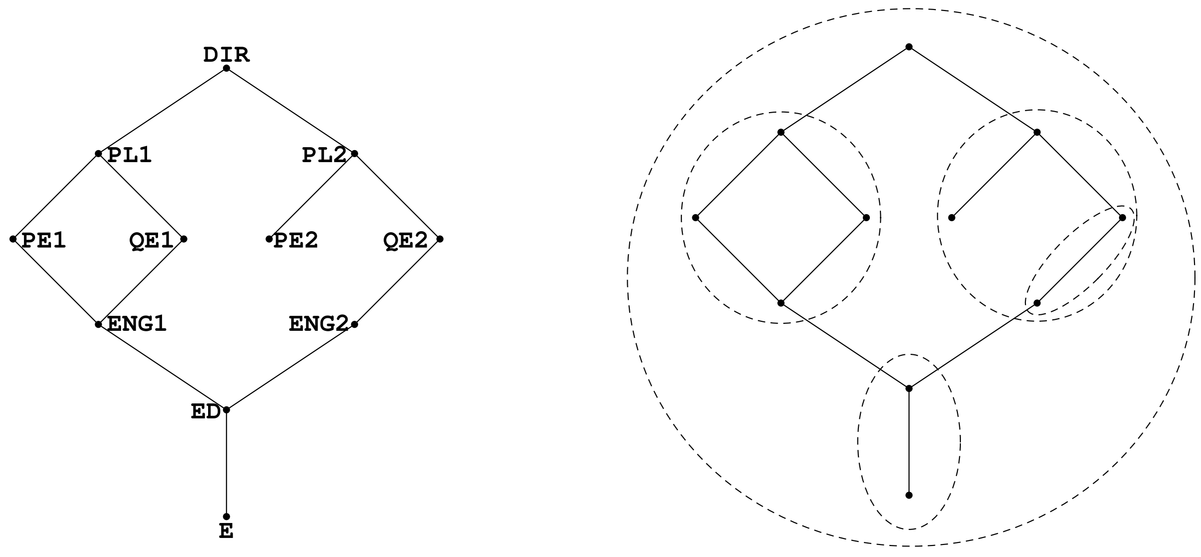
Lemma 2 states that administrative domains are either nested or disjoint.[3] An illustration of this result is given in Figure 1(b); domains are enclosed by broken lines. The hierarchy depicted in Figure 1(a) is adapted from an example by Sandhu. Hence, for any partially ordered set of roles R, the partially ordered set $\langle \mathcal{D}, \subseteq \rangle$ is a tree. Figure 2 illustrates the administrative domain tree (including trivial domains) for the role hierarchy depicted in Figure 1.

The administrative domain tree provides a natural ordering on the set of administrators. Specifically, if a and b are administrators, we write $a \preccurlyeq b$ if $\sigma(a) \subseteq \sigma(b)$.

A corollary of Lemma 2 is that for every role $r \in R$, there exists a smallest (non-trivial) administrative domain to which r belongs, which we will denote by $[r]$.[4] Since $[r]$ is an administrative domain, $[r] = \sigma(a)$ for some role a, and we will say that a is the *line manager* of role r. From Figure 2 we see that $[\texttt{PE1}] = \{\texttt{ENG1}, \texttt{PE1}, \texttt{QE1}, \texttt{PL1}\}$, for example, and hence that `PL1` is the line manager of `PE1`. Let $X \subseteq R$. We define $\lfloor X \rfloor$ to be the largest administrative domain D such that $D \subseteq [x]$ for all $x \in X$, and $\lceil X \rceil$ to be the smallest administrative domain such that $[x] \subseteq D$ for all $x \in X$.We have, for example, $\lfloor \texttt{QE2}, \texttt{PL2} \rfloor = [\texttt{QE2}]$ and $\lceil \texttt{QE2}, \texttt{PL2} \rceil = [\texttt{PL2}]$; and $\lfloor \texttt{QE1}, \texttt{PL2} \rfloor = \emptyset$ and $\lceil \texttt{QE1}, \texttt{PL2} \rceil = R$. Note that if there exist $x, y \in X$ such that $[x] \cap [y] = \emptyset$ then $\lfloor X \rfloor = \emptyset$.

[3]ARBAC97 introduces the concept of *authority ranges* which are defined by the administrator of the system. It is required that authority ranges are either nested or disjoint. It is interesting that this property "comes for free" with administrative scope. We will consider this in more detail in Section 6.

[4]This domain is simply the (unique) immediate parent of r in the domain tree. ARBAC97 defines the concept of an *immediate authority range*, which is analogous to this type of administrative domain.



(a) Role names (b) Administrative domains

Figure 1: An example role hierarchy

4 Preserving administrative scope

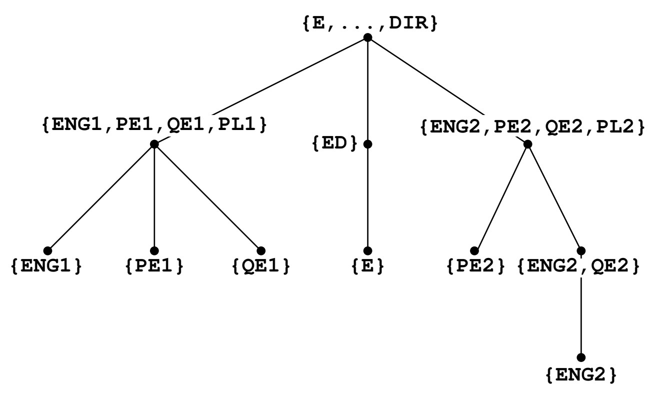
In Section 2.3 we observed that the effect of a hierarchy operation is not necessarily limited to the parameters of the operation. A consequence of this is that the administrative scope of a role can change following a hierarchy operation. If, for example, PL1 deletes the edge $(\texttt{PE1}, \texttt{PL1})$, a new edge $(\texttt{PE1}, \texttt{DIR})$ is added to preserve inheritance and PE1 no longer belongs to $\sigma(\texttt{PL1})$.

Of course, these operational semantics may be regarded as acceptable in certain situations. However, if we assume that it is desirable for a hierarchy operation to preserve administrative scope, then it is necessary to impose some additional conditions that must be satisfied if the operation is to succeed. There are at least three different possibilities. Specifically, if a performs a hierarchy operation we could require that:

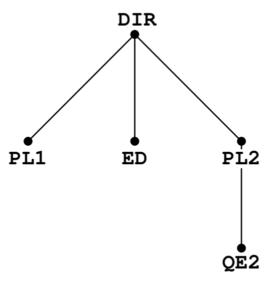
- $\sigma(a)$ should be preserved;
- $\sigma(a')$ should be preserved for all $a' \geqslant a$;
- $\sigma(a')$ should be preserved for all a'.

4.1 Scope preserving hierarchy operations

A hierarchy operation may cause a change to R or the partial ordering defined on R. If $S \subseteq R$, we will write S' or $(S)'$ to denote the value of S following a hierarchy operation.



(a) Administrative domains



(b) Administrators

Figure 2: The administrative domain tree for the role hierarchy in Figure 1

In particular, we will write $\sigma(a)'$ to denote the administrative scope of a following an operation, but for clarity we prefer to write $(\uparrow x)'$, $(\downarrow x)'$ and $(\updownarrow x)'$ (rather than $\uparrow x'$, $\downarrow x'$ and $\updownarrow x'$). Informally, we say S is preserved by a hierarchy operation if anything in S prior to the operation remains in S if it remains in R. More formally, we have the following definition.

Definition 4 *Let $S \subseteq R$. We say S is* preserved *by a hierarchy operation if $S \cap R' \subseteq S'$.*

Definition 5 *We say an operation performed by a is*

- locally scope preserving *if it preserves $\sigma(a)$;*
- hierarchically scope preserving *if it preserves $\sigma(b)$ for all $b \in R$ such that $\sigma(a) \subseteq \sigma(b)$;*
- universally scope preserving *if it preserves $\sigma(b)$ for all $b \in R$.*

For convenience, we will say an operation is 0SP if it is always locally scope preserving, 1SP if it is always hierarchically scope preserving, and 2SP if it is always universally scope preserving. It is clear from the definition that if an operation is 2SP then it is also 0SP and 1SP, and that if an operation is 1SP then it is also 0SP.

Note that hierarchy operations are not, in general, 0SP. The operation deleteEdge(PL1, PE1, PL1) defined in Table 2 is not 0SP, since $\sigma(\texttt{PL1}) = \{\texttt{ENG1}, \texttt{QE1}, \texttt{PE1}, \texttt{PL1}\}$ and $\sigma(\texttt{PL1})' = \{\texttt{QE1}, \texttt{PL1}\}$. Hence it is necessary to impose restrictions on the hierarchy operations that are permitted to succeed (if we wish to preserve administrative scope). We address these issues in the next section and also specify conditions that define 0SP, 1SP and 2SP operations.

Definition 6 *We say an operation performed by a is* autonomy preserving *if there does not exist $b \leqslant a$ such that b is permitted to perform the same operation.*

We say an operation is 3SP if it is autonomy preserving. An example will make this notion clear: let a and b be administrators with $\sigma(b) \subseteq \sigma(a)$ and $r \in \sigma(b)$; then deleteRole(a, r) succeeds if the operation is 2SP but fails if it is 3SP. In other words, a 3SP operation will only succeed if it is invoked by the most local administrator: senior administrators cannot change nested administrative domains within their scope.

4.2 Scope preserving administrative models

An administrative model $\mathcal{M}$ is part of the reference monitor that determines whether requests to perform administrative operations should succeed. Typically, $\mathcal{M}$ specifies conditions for each hierarchy operation that must be satisfied for that operation to succeed (as in Table 2, for example). We say a hierarchy operation is *$\mathcal{M}$-permissible* if the condition(s) permit the operation to proceed. Some conditions may only preserve the administrative scope of the role that performs the operation, while others may preserve the administrative scope of all roles. We now introduce a classification scheme for administrative models by extending the definitions of 0SP, 1SP, 2SP and 3SP for hierarchy operations in the natural way.

Definition 7 *We say that $\mathcal{M}$ is iSP if all $\mathcal{M}$-permissible hierarchy operations are iSP, $0 \leqslant i \leqslant 3$.*

The RHA family of models is not 0SP. This is a potential criticism of the RHA family of models, although it should be noted that an administrative role can never increase its own administrative scope by performing a hierarchy operation. Nevertheless, we believe this provides sufficient motivation for introducing the idea of 0SP.

Informally, we note that one problem with the set of conditions in Table 2 is that deleting an edge can "break" the administrative scope of the role performing the deletion. This problem arises because the range of the operation includes roles outside the administrative scope of the role performing the deletion. In the case of the operation deleteEdge$(\texttt{PL1}, \texttt{PE1}, \texttt{PL1})$, the range of the operation includes $\texttt{DIR} \notin \sigma(\texttt{PL1})$.

However, a 0SP model does not necessarily prevent a role a from performing a hierarchy operation that preserves $\sigma(a)$ but does not preserve the administrative scope of a more senior role. In many situations, we would not want this to happen, hence the idea of 1SP models.

Note that an 1SP model would permit the operation addRole$(\texttt{DIR}, \{\texttt{QE1}\}, \{\texttt{DIR}\})$, which does not preserve $\sigma(\texttt{PL1})$. As further example, deleteEdge$(\texttt{DIR}, \texttt{ENG1}, \texttt{QE1})$ is 2SP, but deleteEdge$(\texttt{DIR}, \texttt{QE1}, \texttt{PL1})$ is not. There may be situations – when we wish to guarantee the autonomy of administrative domains, for example – where we want the administrative scope of every role to be preserved by every hierarchy operation; hence the introduction of 2SP models. (We shall see later that ARBAC97 is approximately 2SP, although it was never characterized in this way when it was introduced.)

Finally, we note that an 2SP model would permit the operation deleteRole($\mathtt{DIR}, \mathtt{QE1}$). Although this operation preserves $\sigma(\mathtt{PL1})$, since $\sigma(\mathtt{PL1}) \cap R' = \sigma(\mathtt{PL1})'$, we may wish to strengthen the autonomy of domains by preventing more senior administrators changing nested domains and hence we introduce the idea of 3SP models.

Table 3 lists four different sets of conditions that must be satisfied for hierarchy operations to be successful. We will use these sets in the remainder of this section to prove the existence of 0SP, 1SP, 2SP and 3SP administrative models: $\mathcal{C}_{rha}$ is the set of conditions used by the RHA family of models, and is reproduced from Table 2 for convenience; $\mathcal{C}_i$ gives rise to an iSP model, $i = 0, 2, 3$. We also prove that $\mathcal{C}_0$ is sufficient to define a 1SP model.

Each column in the table specifies a set of conditions for each hierarchy operation. The conditions become increasingly restrictive from left to right. Each set of conditions is derived in part from the previous set. We only highlight the changes from the previous column in order to simplify the table's interpretation. Note the following features of the table:

- A new condition has been introduced in order to make deleteEdge 0SP;
- New conditions are required to define 2SP operations when those operations may add edges to the hierarchy. Informally, the new conditions require that new edges are directed from children within a larger administrative domain to parents in a smaller administrative domain;
- New conditions are required to define 3SP operations. Informally, these conditions require that the most local administrator performs the operation to preserve autonomy.

Operation	$\mathcal{C}_{rha}$	$\mathcal{C}_0$	$\mathcal{C}_2$	$\mathcal{C}_3$
addRole(a, r, C, P)	$C \subseteq \widehat{\sigma}(a)$ $P \subseteq \sigma(a)$	$C \subseteq \widehat{\sigma}(a)$ $P \subseteq \sigma(a)$	$C \subseteq \widehat{\sigma}(a)$ $P \subseteq \sigma(a)$ $\lceil P \rceil \subseteq \lfloor C \rfloor$	$C \subseteq \widehat{\sigma}(a)$ $P \subseteq \sigma(a)$ $[C] = \sigma(a)$
deleteRole(a, r)	$r \in \widehat{\sigma}(a)$	$r \in \widehat{\sigma}(a)$	$r \in \widehat{\sigma}(a)$	$r \in \widehat{\sigma}(a)$ $[r] = \sigma(a)$
addEdge(a, c, p)	$c, p \in \sigma(a)$	$c, p \in \sigma(a)$	$c, p \in \sigma(a)$ $[p] \subseteq [c]$	$c, p \in \sigma(a)$ $[c] = \sigma(a)$
deleteEdge(a, c, p)	$c, p \in \sigma(a)$	$c, p \in \widehat{\sigma}(a)$	$c, p \in \widehat{\sigma}(a)$ $\lceil \nabla p \rceil \subseteq [c]$	$c, p \in \widehat{\sigma}(a)$ $[c] = \sigma(a)$

Table 3: Scope preserving conditions

Henceforth we will write o_x to denote that we are considering hierarchy operation o using conditions $\mathcal{C}_x$ from Table 3. The operation addRole$_2(a, r, C, P)$, for example, only succeeds if $C \subseteq \widehat{\sigma}(a)$, $P \subseteq \sigma(a)$ and $\lceil P \rceil \subseteq \lfloor C \rfloor$.

Remark 8 *It is worth noting that each of the conditions in Table 3 can be easily checked using the domain tree. To check that $[c] = \sigma(a)$, for example, it is simply a matter of confirming that a is the immediate parent of c in the tree.*

Theorem 9 *$\mathcal{C}_0$ is* $\mathsf{0SP}$.

Theorem 10 *$\mathcal{C}_1$ is* $\mathsf{1SP}$.

Theorem 11 *$\mathcal{C}_2$ is* $\mathsf{2SP}$.

Corollary 12 *$\mathcal{C}_3$ is* $\mathsf{2SP}$.

Theorem 13 *$\mathcal{C}_3$ is* $\mathsf{3SP}$.

Proofs of all these results are given in the appendix. We also state and prove two auxiliary results, which are used to prove Theorem 9. Theorem 10 is proved by extending the proof method used for Theorem 9. Theorem 11 is proved using the fact that domains are either nested or disjoint and that $\mathcal{C}_2$ only permits the addition of edges to the hierarchy if they are directed into interior domains, thereby preserving the set of senior roles of the child role. Corollary 12 is established by proving that $\mathcal{C}_3$ implies $\mathcal{C}_2$ and then using Theorem 11, and again makes use of the fact that domains are nested. Theorem 13 follows from the definition of $\mathcal{C}_3$ and a simple proof by contradiction.

4.3 Discussion

There is one unfortunate consequence of Theorems 11 and 13. In particular, there are certain hierarchy operations that cannot be performed by any administrative role in a $\mathsf{2SP}$ or $\mathsf{3SP}$ model. The simplest example is $\mathsf{deleteRole}_3(a, r)$, when r is an administrator. In this case, $a > r$ cannot perform the operation because $\sigma(r) \subset \sigma(a)$ and hence $[r] \neq \sigma(a)$; nor can r delete itself because $r \notin \widehat{\sigma}(r)$. Another example is the operation $\mathsf{deleteEdge}_2(a, \mathtt{ENG2}, \mathtt{QE2})$: $a \neq \mathtt{PL2}$ since $[\triangledown\mathtt{QE2}] \not\subseteq [\mathtt{ENG2}]$; $a \neq \mathtt{QE2}$ since $\mathtt{QE2} \notin \widehat{\sigma}(\mathtt{QE2})$. The developers of ARBAC97 were aware of this problem and solved it simply by explicitly refusing to consider such operations. In this general framework, it is simply the case that these operations cannot be performed.

There are two ways of addressing this problem: these methods can be used separately or in conjunction. The first is to allow $\mathsf{1SP}$ operations in certain situations, although this leads to a non-uniform treatment of hierarchy operations. The alternative is to not use every administrative domain in the role hierarchy. This can be achieved simply by associating certain administrative domains with administrative roles, in the style of ARBAC97. We might, for example, have initially assigned the domain $\sigma(\mathtt{PL1})$ to a project security officer role $\mathtt{PSO1}$ and the domain R to the senior security officer role $\mathtt{SSO}$. If we wish to perform the operation $\mathsf{deleteEdge}(\mathtt{DIR}, \mathtt{QE1}, \mathtt{PL1})$ which is not permitted by a $\mathsf{2SP}$ model, we can simply delete the association between $\mathtt{PSO1}$ and $\sigma(\mathtt{PL1})$ so that $[\mathtt{QE1}] = R$, thereby permitting the deletion of the edge.

In the next section we will explore this option in more detail and develop a general design pattern for role-based administrative models, which we call RBAT (role-based administration framework). We will show how particular instances of the framework are related to RHA_4 and ARBAC97.

5 RBAT: A template for role-based administrative models

We have introduced the idea of an administrative domain and a number of criteria that can be used to control the way in which administrative domains are affected by hierarchy operations. In this section we describe RBAT (role-based administration template), which provides a design pattern for role-based administration models. We will show how particular instances of the framework are related to RHA_4 and ARBAC97.

5.1 Components of RBAT

RBAT defines the following components:

- A non-empty set of domains $\mathcal{D}$, each of which contains a unique administrator role. Moreover, for all $D, D' \in \mathcal{D}$, one of the following conditions must hold: (i) $D \subseteq D'$ (ii) $D \supseteq D'$ (iii) $D \cap D' = \emptyset$;
- A set of administrative operations $\mathcal{O}$;
- A set of conditions $\mathcal{C}$, each of which determines the success of a particular operation;
- A set of administrative roles R_A, which may be empty;
- A relation $\texttt{can-administer} \subseteq R_A \times R$, which associates an administrative role with the administrator of a domain. If $R_A = \emptyset$, $\texttt{can-administer} \subseteq R \times R$.

5.2 The `can-administer` relation

Instead of using roles in the hierarchy, we may define a distinct set of administrative roles and assign them to administrative domains within the role hierarchy. This is similar to the approach taken in ARBAC97 and is a simplification of the `admin-auth` relation in the RHA family of models.

Since an administrative domain is uniquely determined by its administrator, we can introduce a relation $\texttt{can-administer} \subseteq R_A \times R$, where R_A is the set of administrative roles. The semantics of $(a, r) \in \texttt{can-administer}$ is that a has administrative control of $\sigma(r)$, the administrative domain defined by r. Hence $(\texttt{PSO1}, \texttt{PL1})$, for example, could be used to specify that `PSO1` has been granted administrative control over the domain $\sigma(\texttt{PL1}) = \{\texttt{ENG1}, \texttt{PE1}, \texttt{QE1}, \texttt{PL1}\}$.

Table 4 shows the conditions for success of hierarchy operations within this general framework. In simple terms, an operation performed by an administrative role a succeeds if all the arguments of the operation belong to a single administrative domain $\sigma(x)$ that is controlled by a. The model can be chosen to be 1SP, 2SP or 3SP, simply by selecting the appropriate criteria for the operation to succeed when performed by x.

Operation	**Conditions**	
$\mathsf{addRole}(a, r, C, P)$	$\exists x \in R,\ (a, x) \in \texttt{can-administer}$	$\mathsf{addRole}(x, r, C, P)$ succeeds
$\mathsf{deleteRole}(a, r)$		$\mathsf{deleteRole}(x, r)$ succeeds
$\mathsf{addEdge}(a, c, p)$		$\mathsf{addEdge}(x, c, p)$ succeeds
$\mathsf{deleteEdge}(a, c, p)$		$\mathsf{deleteEdge}(x, c, p)$ succeeds

Table 4: Success of hierarchy operations in RBAT

5.3 The RHA$_4$ model

It is natural to expect that the model we have derived has some similarity with our original RHA family of models, since administrative domains are synonymous with administrative scope. Note that RHA$_1$ is a special case of RBAT in which $R_A = \emptyset$, $\texttt{can-administer} = \{(r, r) : r \in R\}$ and $\mathcal{C} = \mathcal{C}_{rha}$.

The `can-administer` relation is identical in structure to the relation $\texttt{admin-auth} \subseteq R_A \times R$ defined in RHA$_4$, the most complex model of the RHA family. In RBAT, we define the administrative scope of an administrative role a to be the union of the domains it controls and insist that for any command to succeed, all arguments must belong to a single one of those domains. However, in RHA$_4$, the administrative scope of a was defined in terms of the roles controlled by a (that is, $\{r \in R : (a, r) \in \texttt{admin-auth}\}$). An example should make the difference clearer: the RHA model would permit $(\texttt{PSO1}, \texttt{PE1}), (\texttt{PSO1}, \texttt{QE1}) \in \texttt{admin-auth}$, meaning that $\sigma(\texttt{PSO1}) = \{\texttt{ENG1}, \texttt{PE1}, \texttt{QE1}\}$, whereas these pairs are not permitted in the `can-administer` relation because PE1 and QE1 are not administrators in R. Moreover, although we permit $(\texttt{PSO1}, \texttt{PL1}), (\texttt{PSO1}, \texttt{PL2}) \in \texttt{can-administer}$, for example, we do not permit the operation $\mathsf{addEdge}_2(\texttt{PSO1}, \texttt{ENG1}, \texttt{QE2})$. Strictly speaking, then, RHA$_4$ is not an instance of RBAT, although an 0SP model is a close approximation to RHA$_4$.

6 Connections with ARBAC97

What is more surprising is that the ARBAC97 model can be expressed in terms of the framework described in the last section. ARBAC97 defines the relation $\texttt{can-modify} \subseteq R_A \times \mathcal{E}$, where $\mathcal{E}$ is the set of encapsulated ranges in R (see Definition 14 below). Roughly speaking, the administrative role $a \in R_A$ can perform a hierarchy operation provided the arguments are contained in some encapsulated range E and $(a, E) \in \texttt{can-modify}$. In addition, no hierarchy operation may violate the encapsulation of the ranges in contained in the `can-modify` relation: this is clearly a kind of preservation property.

In this section we identify a strong link between encapsulated ranges and administrative domains and provide a new formulation of ARBAC97. We also identify a couple of weaknesses in the original formulation, which become apparent when the ARBAC97 model is interpreted within our framework.

The following definition is due to Sandhu *et al* [11, Definition 16], although it has been slightly modified as a result of an observation made by Crampton and Loizou [4, Remark 7.3].

Definition 14 *A range* $[x, y]$ *is* encapsulated *if for all* $z \in (x, y)$ *and all* $w \notin (x, y)$*:*

$$w > z \text{ iff } w \geqslant y;$$
$$w < z \text{ iff } w \leqslant x.$$

Proposition 15 *Every encapsulated range is an administrative domain.*

Proof Let $[x, y]$ be an encapsulated range and let $z \in [x, y]$. We will show that $\uparrow z \subseteq \updownarrow y$. Now $\uparrow z = \{w \in R : w \geqslant z\}$. Then $z \in \downarrow y$ if $z \leqslant w \leqslant y$. Otherwise we have $z < w$ and, since $[x, y]$ is encapsulated, $w \geqslant y$ and $w \in \uparrow y$. The result follows. ■

Corollary 16 *Encapsulated ranges are either nested or disjoint.*

Proof The result follows immediately from Lemma 2 and Proposition 15. ■

The converse of Proposition 15 is not true because an administrative domain is not necessarily a range. However, we have the following definition and result.

Definition 17 *An* administrative range *is a range* $[b, t] \in R$ *such that for all* $x \in [b, t]$, $\uparrow x \subseteq \updownarrow t$ *and* $\downarrow x \subseteq \updownarrow b$.[5]

Note that the definition of administrative range is the symmetric analogue of the definition of administrative scope. We will use this fact later when deriving conditions for an operation to preserve encapsulated ranges.

Proposition 18 *The range* (b, t) *is encapsulated iff* $[b, t]$ *is an administrative range.*

Proof The result follows from the definition of administrative range and the proof method of Proposition 15. ■

[5] b denotes "bottom" and t denotes "top".

Note that an encapsulated range does not include the end points that define it. In other words, an encapsulated range is analogous to strict administrative scope (which omits the top element in an administrative domain). To avoid the introduction of any further notation, we will write $\widehat{\sigma}(a)$ to denote the encapsulated range with top element a.

In ARBAC97, the ranges that appear in the `can-modify` relation are called *authority ranges*. They are defined by the system administrator and are required to be encapsulated ranges. Moreover, it is required that each pair of authority ranges be either nested or disjoint. Corollary 16 shows that this requirement is redundant as encapsulated ranges are either nested or disjoint by definition.

The success of many operations in ARBAC97 depends on the notion of an *immediate authority range*. Since authority ranges are nested or disjoint by definition, there exists a smallest authority range to which any given role belongs. The immediate authority range of a role r is analogous to $[r]$.

We now place ARBAC97 in the context of the framework developed in this paper. In ARBAC97, every hierarchy operation must preserve the encapsulation of all authority ranges. The designers of ARBAC97 give no rules or method for determining whether a hierarchy operation satisfies this condition. It should come as no surprise by now that we are able to express ARBAC97 using the approach described in the previous section and that we can explicitly state sufficient conditions for an operation to preserve encapsulated ranges. Specifically, $\mathcal{D} = \mathcal{E}$, $\mathcal{C} = \mathcal{C}_2$, and `can-modify` = `can-administer`.

In Table 5 we summarize the conditions that must be satisfied for a hierarchy operation to succeed in the ARBAC97 model. We write $x \in \sigma(a)$ as an abbreviation for "there exists $r \in R$ such that $x \in \sigma(r)$ and $(a, r) \in$ `can-modify`". Expressions such as $X \subseteq \sigma(a)$ and $x \in \widehat{\sigma}(a)$ have analogous interpretations.

The second column restates the conditions given by Sandhu in the original formulation of the model. Notice the use of $\widehat{\sigma}(a)$ in the second column, corresponding to the fact that the basic unit of administration in ARBAC97 is the encapsulated range, which does not include the end points of the range. We have simplified some of the conditions for the addRole and addEdge operations, which were permitted in the original formulation of ARBAC97 provided one of three conditions was satisfied, one of which was that $[c] = [p]$. In fact, each of these conditions turns out to be equivalent. By symmetry, and using $\mathcal{C}_2$ from Table 3, all encapsulated ranges are preserved if $[c] \subseteq [p]$ and $[p] \subseteq [c]$. Therefore, addRole$(a, r, \{c\}, \{p\})$ succeeds if $c, p \in \widehat{\sigma}(a)$ and $[c] = [p]$.

The third column suggests some slight modifications to these conditions that should yield an improved version of ARBAC97. Specifically, we add a condition to the deleteEdge operation that guarantees that all such operations preserve encapsulated ranges. Sandhu *et al* do not comment on the fact that deleting an edge can destroy an encapsulated range and make no effort to prevent this happening. In contrast, we introduced a new requirement into $\mathcal{C}_2$ for the deleteEdge operation in order to take account of this fact. Correspondingly, ARBAC97 should include the following condition for the deleteEdge operation in order to preserve all encapsulated ranges: $[p] = [c]$ since we require that $[\triangledown p] \subseteq [c]$ and $[\triangle c] \subseteq [p]$ in order to preserve encapsulated ranges. We also make the definition of addRole more general, in line with the addRole operation used elsewhere in this paper. The ARBAC97 model requires that a new role have a single child and parent role. There is no theoretical reason for this restriction. Hence we suggest that we use the operation $\mathsf{addRole}(a, r, C, P)$, and that it succeeds if $C \subseteq \widehat{\sigma}(a)$, $P \subseteq \sigma(a)$ and $\lfloor C \rfloor = \lceil C \rceil = \lfloor P \rfloor = \lceil P \rceil$. (This latter condition simply says that there exists b such that for all $c \in C$ and all $p \in P$, $[c] = [p] = \sigma(b)$.)

Operation	**Sandhu** *et al*	**Crampton**
$\mathsf{addRole}(a, r, C, P)$	$C = \{c\}$ $P = \{p\}$ $c, p \in \widehat{\sigma}(a)$ $[c] = [p]$	$C \subseteq \widehat{\sigma}(a)$ $P \subseteq \sigma(a)$ $\lfloor C \rfloor = \lceil C \rceil = \lfloor P \rfloor = \lceil P \rceil$
$\mathsf{deleteRole}(a, r)$	$r \in \widehat{\sigma}(a)$	$r \in \widehat{\sigma}(a)$
$\mathsf{addEdge}(a, c, p)$	$c, p \in \widehat{\sigma}(a)$ $[c] = [p]$	$c, p \in \sigma(a)$ $[c] = [p]$
$\mathsf{deleteEdge}(a, c, p)$	$c, p \in \widehat{\sigma}(a)$	$c, p \in \widehat{\sigma}(a)$ $[c] = [p]$

Table 5: Hierarchy operations in ARBAC97

To conclude this section we provide a comparison of the success of a number of different typical hierarchy operations that could be performed on the hierarchy of Figure 1(a). Figure 3 (on page 20) compares five different models: RHA, $\mathsf{1SP}$, $\mathsf{2SP}$ and $\mathsf{3SP}$ models, and ARBAC97. It is assumed that the changes are not cumulative: that is, each operation is performed on the hierarchy shown in Figure 1(a). (The hierarchy is reproduced in Figure 3 for the reader's convenience.) The operations are listed in decreasing order of the number of models that permit the operation to be performed. Note that each model permits a different subset of the operations to succeed.

7 Conclusion

We have provided a characterization of role-based administrative models based on the extent to which the hierarchy operations permitted by the model preserve administrative domains. This characterization enables us to provide a concise description of ARBAC97 and to identify and correct a number of flaws in the original specification.

The success or otherwise of hierarchy operations is determined by the administrative scope of the role performing the operation. The administrative scope of a role can be determined directly and efficiently from the domain tree. Hence it ought to be possible to produce an implementation of ARBAC97 and a number of other models described in this paper for evaluation purposes. Until now, it was not obvious that such an implementation of ARBAC97 existed, since there was no obvious way of testing the requirement that the encapsulation of all authority ranges be preserved.

RBAT mandates the specification of a set of domains, which must be either pairwise nested or disjoint. The obvious choice for such a set is to use those non-trivial domains defined by administrative scope. However, this is not required, and we may also consider the administration of disconnected hierarchies (such as those defined in ERBAC96 [10] and TRBAC [2]). In this instance, it will be necessary for systems administrators to define the administrative domains (that is, without reference to administrative scope) and then use an appropriate set of conditions to determine the success of hierarchy operations and preserve the integrity of those administrative domains. Of course this will require some modification to the conditions $\mathcal{C}_0$, $\mathcal{C}_2$ and $\mathcal{C}_3$, because in these conditions the notion of preserving administrative domains could be neatly captured using the administrative scope of a role. Instead, the conditions will have to be specified in terms of preserving $\downarrow r$, where r is the administrator of the domain, and $\uparrow x$, where x belongs to the domain of which r is the administrator. This does not appear to present insuperable difficulties. To date no administrative model exists for either ERBAC96 or TRBAC.

We can also build real ARBAC97 systems. This was not obvious previously, since there existed a "chicken and egg" situation, in which the `can-modify` relation was defined in terms of the hierarchy, but the `can-modify` relation controlled changes to the hierarchy [4, Section 8.3]. We know that we can build RHA systems if we assume the existence of a system administrator role that initially controls an empty domain [4], which therefore suggests that we can also build ARBAC97 systems.

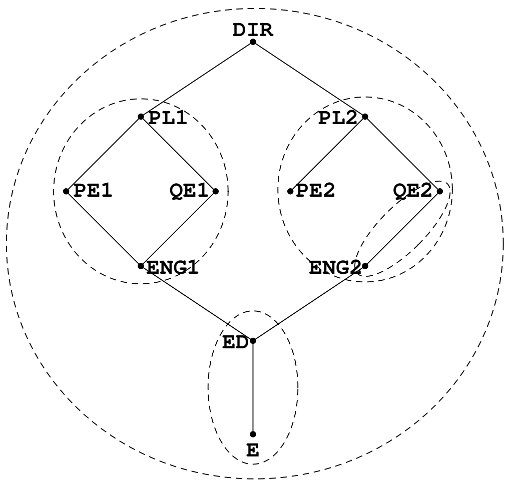
Moreover, we can define more "relaxed" ARBAC97-style models. The original version of the model is essentially an 2SP model, since it requires that all encapsulated ranges be preserved. We now have a framework that enables to develop a set of less restrictive models based on authority ranges, but with weaker preservation properties such as 0SP.

Finally, we note that this work may have a considerable impact on the study of the safety problem in role-based systems [3, 7, 8]. The safety problem considers the propagation of access rights due to changes to access control data structures and hence every administrative model gives rise to an instance of the safety problem. Since we can now order administrative models according to the extent to which they preserve domains, if we know that safety problem is undecidable in one model, then we conjecture that it is undecidable in any more permissive model. We anticipate that domain preservation will be a feature of models for which the safety problem is decidable in polynomial time.

We believe that this work will be of benefit to application and systems developers, who wish to understand the mechanisms of role-based administration better and to know what properties will be preserved by the administrative model they choose to implement. We also believe this work lays a valuable theoretical foundation for the further development of role-based administrative models and investigation of the safety problem in role-based systems.

Future work will include the construction of administrative models for ERBAC96 and TRBAC as suggested above. We will also extend RBAT to include all administrative operations (as defined in Section 2) and the administration of the `can-administer` relation. A further interesting possibility is to introduce administrative permissions. This leads to a two-phase checking process for administrative operations, similar to that in the Bell-

LaPadula model: the operation should be both explicitly permitted by the assignment of appropriate administrative permissions and should satisfy the conditions for the operation to proceed. In this context, the conditions form the mandatory element of the access control checking process and the permissions form the discretionary element. The introduction of administrative permissions also suggests the possibility of administrative separation of duty. We could, for example, insist that for a given administrative domain, a human resources role is responsible for the administration of the user-role assignment relation, whereas some managerial or systems administrator role is responsible for the administration of the permission-role assignment relation. Finally, we hope to investigate the notion of role "visibility", which we regard as being analogous to the scope of a variable in block structured programming languages. Some roles and edges within a domain might only be visible to roles within that domain, whereas other might have global visibility. This concept may help to address the fact that it is not always appropriate for all permissions to be available to all more senior roles. Certainly, there is no shortage of directions in which this work can be developed.



(a) Role hierarchy

Operation	$\mathcal{C}_{rha}$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_{arbac}$
deleteEdge(PSO1, ENG1, QE1)	✓	✓	✓	✓	✓
deleteRole(PSO1, PE1)	✓	✓	✓	✓	✓
deleteRole(SSO, PE1)	✓	✓	✓	✗	✓
addRole(PSO1, Y, $\emptyset$, {PE1})	✓	✓	✓	✓	✗
addRole(PSO1, Z, {PE1, QE1}, $\emptyset$)	✓	✓	✓	✓	✗
addRole(SSO, W, {ED}, {PE1})	✓	✓	✓	✓	✗
deleteRole(PSO1, ENG1)	✓	✓	✓	✓	✗
addEdge(SSO, ED, PE2)	✓	✓	✓	✓	✗
deleteEdge(SSO, ED, ENG1)	✓	✓	✓	✓	✗
deleteEdge(SSO, PE1, PL1)	✓	✓	✓	✗	✗
addRole(SSO, X, {QE1}, {DIR})	✓	✓	✗	✗	✗
addRole(SSO, V, {ENG1}, {PE2})	✓	✓	✗	✗	✗
addEdge(SSO, ENG1, PE2)	✓	✓	✗	✗	✗
deleteEdge(PSO1, PE1, PL1)	✓	✗	✗	✗	✗
addRole(PSO1, W, {ED}, {PE1})	✗	✗	✗	✗	✗
addRole(PSO1, V, {ENG1}, {PE2})	✗	✗	✗	✗	✗
addEdge(PSO1, ENG1, PE2)	✗	✗	✗	✗	✗

(b) Operations

Figure 3: The success of hierarchy operations in different administrative models: It is assumed that (PSO1, PL1), (SSO, DIR) $\in$ `can-administer`

References

[1] American National Standards Institute. *ANSI INCITS 359-2004 for Role Based Access Control*, 2004.

[2] E. Bertino, P.A. Bonatti, and E. Ferrari. TRBAC: A temporal role-based access control model. *ACM Transactions on Information and System Security*, 4(3):191–223, 2001.

[3] J. Crampton. *Authorization and antichains*. PhD thesis, Birkbeck, University of London, London, England, 2002. Available from `http://www.isg.rhul.ac.uk/~jason`.

[4] J. Crampton and G. Loizou. Administrative scope: A foundation for role-based administrative models. *ACM Transactions on Information and System Security*, 6(2):201–231, 2003.

[5] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.

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

[7] N. Li and M.V. Tripunitara. Security analysis in role-based access control. In *Proceedings of the Ninth ACM Symposium on Access Control Models and Technologies*, pages 126–135, 2004.

[8] Q. Munawer and R. Sandhu. Simulation of the augmented typed access matrix model (ATAM) using roles. In *Proceedings INFOSECU99 International Conference on Information Security*, 1999.

[9] M. Nyanchama and S. Osborn. The role graph model and conflict of interest. *ACM Transactions on Information and System Security*, 2(1):3–33, 1999.

[10] R. Sandhu. Role activation hierarchies. In *Proceedings of Third ACM Workshop on Role-Based Access Control*, pages 33–40, 1998.

[11] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security*, 1(2):105–135, 1999.

[12] R. Sandhu, E.J. Coyne, H. Feinstein, and C.E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

A Proofs

We first prove two preliminary results.

Proposition 19 *$\downarrow a$ and $\uparrow a$ are preserved by all hierarchy operations o_{rha} performed by a, with the exception of* $\mathsf{deleteEdge}_{rha}(a, c, a)$.

Proof of Proposition 19 We first note the following state changes effected by hierarchy operations, which follow from the operational semantics given in Table 1:

$$\mathsf{addEdge}(a, c, p) \quad \Rightarrow \quad (\uparrow x)' = \begin{cases} \uparrow x \cup \uparrow p & \text{if } x \in \downarrow c, \\ \uparrow x & \text{otherwise;} \end{cases} \tag{1}$$

$$(\downarrow x)' = \begin{cases} \downarrow x \cup \downarrow c & \text{if } x \in \uparrow p, \\ \downarrow x & \text{otherwise;} \end{cases} \tag{2}$$

$$\mathsf{deleteEdge}(a, c, p) \quad \Rightarrow \quad (\uparrow x)' = \begin{cases} \uparrow x \setminus \{p\} & \text{if } x = c, \\ \uparrow x & \text{otherwise;} \end{cases} \tag{3}$$

$$(\downarrow x)' = \begin{cases} \downarrow x \setminus \{c\} & \text{if } x = p, \\ \downarrow x & \text{otherwise;} \end{cases} \tag{4}$$

$$\mathsf{addRole}(a, r, C, P) \quad \Rightarrow \quad (\uparrow x)' = \begin{cases} \uparrow x \cup \{r\} \cup \uparrow P & \text{if } x \in \downarrow C, \\ \uparrow x & \text{otherwise;} \end{cases} \tag{5}$$

$$(\downarrow x)' = \begin{cases} \downarrow x \cup \{r\} \cup \downarrow C & \text{if } x \in \uparrow P, \\ \downarrow x & \text{otherwise;} \end{cases} \tag{6}$$

$$\mathsf{deleteRole}(a, r) \quad \Rightarrow \quad (\uparrow x)' = \begin{cases} \uparrow x \setminus \{r\} & \text{if } x < r, \\ \uparrow x & \text{otherwise;} \end{cases} \tag{7}$$

$$(\downarrow x)' = \begin{cases} \downarrow x \setminus \{r\} & \text{if } x > r, \\ \downarrow x & \text{otherwise.} \end{cases} \tag{8}$$

The result for $\uparrow a$ follows automatically from the fact that any argument in any operation must belong to $\sigma(a)$ and hence is less than or equal to a. This means that a hierarchy operation performed by a cannot affect $\uparrow a$.

$\mathsf{deleteRole}_{rha}(a, r)$ preserves $\downarrow a$ since $r < a$ and hence $(\downarrow a)' = \downarrow a \setminus \{r\} = \downarrow a \cap R'$. Following the $\mathsf{addEdge}_{rha}(a, c, p)$ operation we have $(\downarrow a)' = \downarrow a \cup \downarrow c$, but since $c \in \sigma(a)$, we have $c \leqslant a$ and $(\downarrow a)' = \downarrow a$. A similar argument shows that $\mathsf{addRole}_{rha}$ preserves $\downarrow a$. Finally, if $c \lessdot p < a$ and $p \in \sigma(a)$, there exists $x \in \sigma(a)$ such that $c \lessdot p \lessdot x \leqslant a$, and following $\mathsf{deleteEdge}(a, c, p)$ the edge (c, x) will be added, thereby preserving $\downarrow a$. Hence a problem only arises with the operation $\mathsf{deleteEdge}_{rha}(a, c, a)$, when $(\downarrow a)' = \downarrow a \setminus \{c\}$. ■

Corollary 20 $\mathsf{deleteEdge}_{rha}(a, c, p)$ *preserves* $\downarrow a$ *provided* $p \in \widehat{\sigma}(a)$.

Proof of Theorem 9 By Proposition 19 and Corollary 20, $\downarrow a$ and $\uparrow a$ are preserved by any hierarchy operation o_{rha} performed by a. Let $x \in \sigma(a)$. Hence we need to prove that $(\uparrow x)' \subseteq (\updownarrow a)' = \updownarrow a$. We consider each hierarchy operation in turn.

$$\begin{aligned}
\mathsf{addEdge}_0(a, c, p) \quad &\Rightarrow \quad (\uparrow x)' = \uparrow x \cup \uparrow p && \text{if } x \leqslant c, \\
& \subseteq \uparrow x \cup \updownarrow a && \text{since } p \in \sigma(a), \\
& \subseteq \updownarrow a && \text{since } x \in \sigma(a); \\
\mathsf{deleteEdge}_0(a, c, p) \quad &\Rightarrow \quad (\uparrow x)' \subseteq \uparrow x && \text{since } x \leqslant c < p, \\
& \subseteq \updownarrow a && \text{since } x \in \sigma(a); \\
\mathsf{addRole}_0(a, r, C, P) \quad &\Rightarrow \quad (\uparrow x)' = \uparrow x \cup \{r\} \cup \uparrow P && \text{if } x \in \downarrow C, \\
& \subseteq \uparrow x \cup \updownarrow a && \text{since } P \subseteq \sigma(a), \\
& \subseteq \updownarrow a && \text{since } x \in \sigma(a); \\
\mathsf{deleteRole}_0(a, r) \quad &\Rightarrow \quad (\uparrow x)' \subseteq \uparrow x && \text{since } x \leqslant r, \\
& \subseteq \updownarrow a && \text{since } x \in \sigma(a).
\end{aligned}$$

Hence, $x \in \sigma(a)'$ for each operation and $\mathcal{C}_0$ is $\mathsf{0SP}$. ■

Proof of Theorem 10 Let $b \in R$ such that $\sigma(a) \subseteq \sigma(b)$. We first note that if a performs a hierarchy operation, then $\uparrow b$ and $\downarrow b$ are preserved for all $b \in R$ such that $\sigma(a) \subseteq \sigma(b)$. Clearly, $\uparrow b$ is preserved, because for all $x \in \sigma(a)$, $x \leqslant b$, and therefore changes made by a cannot affect $\uparrow b$. Moreover, $\mathsf{addEdge}_0(a, c, p)$ will not affect $\downarrow b$ since $c, p \in \sigma(b)$ (and hence $c, p \leqslant b$). It is also clear that $\mathsf{addRole}_0$ (respectively $\mathsf{deleteRole}_0$) will not affect $\downarrow b$ except to add (delete) the role from $\downarrow b$. Finally we note $\mathsf{deleteEdge}_0(a, c, p)$ does not change $\downarrow b$ because $p < a \leqslant b$; hence there exists $x \in R$ such that $p \lessdot x \leqslant a \leqslant b$ and hence (c, x) is added to the hierarchy and c remains in $\downarrow b$.

Let $x \in \sigma(b)$. Hence we need to prove that $(\uparrow x)' \subseteq (\updownarrow b)' = \updownarrow b$. We consider each hierarchy operation in turn.

$$\begin{array}{rcll}
\mathsf{addEdge}_0(a,c,p) & \Rightarrow & (\uparrow x)' = \uparrow x \cup \uparrow p & \text{if } x \leqslant c, \\
 & & \subseteq \uparrow x \cup \updownarrow a & \text{since } p \in \sigma(a), \\
 & & \subseteq \uparrow x \cup \updownarrow b & \text{since } a \in \sigma(b), \\
 & & \subseteq \updownarrow b & \text{since } x \in \sigma(b); \\
\mathsf{deleteEdge}_0(a,c,p) & \Rightarrow & (\uparrow x)' \subseteq \uparrow x & \text{since } x \leqslant c < p, \\
 & & \subseteq \updownarrow b & \text{since } x \in \sigma(b); \\
\mathsf{addRole}_0(a,r,C,P) & \Rightarrow & (\uparrow x)' = \uparrow x \cup \{r\} \cup \uparrow P & \text{if } x \in \downarrow C, \\
 & & \subseteq \uparrow x \cup \updownarrow a & \text{since } P \subseteq \sigma(a), \\
 & & \subseteq \uparrow x \cup \updownarrow b & \text{since } a \in \sigma(b), \\
 & & \subseteq \updownarrow b & \text{since } x \in \sigma(b); \\
\mathsf{deleteRole}_0(a,r) & \Rightarrow & (\uparrow x)' \subseteq \uparrow x & \text{since } x \leqslant r, \\
 & & \subseteq \updownarrow b & \text{since } x \in \sigma(b).
\end{array}$$

Hence, $x \in \sigma(b)'$ for each operation and $\mathcal{C}_0$ is 1SP. ■

Proof of Theorem 11 Let $b \in R$ and let $x \in \sigma(b)$. We need to establish that $(\uparrow x)' \subseteq (\updownarrow b)'$.

Consider the hierarchy operation $\mathsf{addEdge}_2(a,c,p)$. Now $[c] = \sigma(a_c)$ and $[p] = \sigma(a_p)$, for some $a_c, a_p \in R$. Since $c, p \in \sigma(a)$ and $[p] \subseteq [c]$, we have $\sigma(a_p) \subseteq \sigma(a_c) \subseteq \sigma(a)$. Recall that by (1) we only need to consider $x \leqslant c$. Since $c \in \uparrow x$ and $x \in \sigma(b)$, we have either (i) $c \in \sigma(b)$ or (ii) $b < c$.

In case (i) we have

$$\begin{array}{rll}
(\uparrow x)' & \subseteq \uparrow x \cup \uparrow p & \text{since } x < c < p, \\
 & \subseteq \uparrow x \cup \updownarrow a_p & \text{since } p \in \sigma(a_p), \\
 & \subseteq \uparrow x \cup \updownarrow a_c & \text{since } p \in \sigma(a_c), \\
 & \subseteq \uparrow x \cup \updownarrow b & \text{since } c \in \sigma(b), \\
 & \subseteq \updownarrow b & \text{since } x \in \sigma(b).
\end{array}$$

In case (ii) we have $b < c < p$ and therefore $(\uparrow b)' = \uparrow b \cup \uparrow p$. Hence

$$\begin{array}{rll}
(\uparrow x)' & \subseteq \uparrow x \cup \uparrow p & \text{since } x < c < p, \\
 & \subseteq \updownarrow b \cup \uparrow p & \text{since } x \in \sigma(b), \\
 & \subseteq (\updownarrow b)' & \text{since } (\uparrow b)' = \uparrow b \cup \uparrow p.
\end{array}$$

Suppose, in order to obtain a contradiction, that the hierarchy operation $\mathsf{deleteEdge}_2(a,c,p)$ is not 2SP. Then there exists an administrator b such that $\sigma(b)$ is not preserved. From Proposition 19, we know a problem will only arise if $p = b$ for some administrator b. In this case, we have $[c] \subseteq \sigma(b) = [p] \subset \lceil \triangledown p \rceil$, which is the desired contradiction (since it is assumed that $\lceil \triangledown p \rceil \subseteq [c]$).

Consider the hierarchy operation $\mathsf{addRole}_2(a, r, C, P)$. If $C = \emptyset$, $(\uparrow x)' = \uparrow x$ for all x, since r has no children. Otherwise, $\lfloor C \rfloor = \sigma(a_C)$ and $\lceil P \rceil = \sigma(a_P)$, for some $a_C, a_P \in R$, and since $C \subseteq \widehat{\sigma}(a)$, $P \subseteq \sigma(a)$ and $\lceil P \rceil \subseteq \lfloor C \rfloor$, we have $\sigma(a_P) \subseteq \sigma(a_C) \subseteq \sigma(a)$. Recall that by (5) we only need to consider $x \in \downarrow C$. Since $x \in \downarrow C$, there exists $c \in C$ such that $c \in \uparrow x$, and since $x \in \sigma(b)$, we have either (i) $c \in \sigma(b)$ or (ii) $b < c$. In case (i) we have

$$\begin{aligned}
(\uparrow x)' &\subseteq \uparrow x \cup \{r\} \cup \uparrow P && \text{since } x < c < p, \text{ for all } p \in P,\\
&\subseteq \uparrow x \cup \{r\} \cup \updownarrow a_p && \text{since } p \in \sigma(a_P), \text{ for all } p \in P,\\
&\subseteq \uparrow x \cup \{r\} \cup \updownarrow a_c && \text{since } p \in \sigma(a_C), \text{ for all } p \in P,\\
&\subseteq \uparrow x \cup \{r\} \cup \updownarrow b && \text{since } c \in \sigma(b),\\
&\subseteq \updownarrow b \cup \{r\} && \text{since } x \in \sigma(b).
\end{aligned}$$

Hence for all $x \in \sigma(b) \cap R'$, $x \in \sigma(b)'$. Note that in this case, $r \in \sigma(b)$.

In case (ii) we have $b < c < r < p$, for all $p \in P$, and therefore $(\uparrow b)' = \uparrow b \cup \{r\} \cup \uparrow P$. Hence

$$\begin{aligned}
(\uparrow x)' &\subseteq \uparrow x \cup \{r\} \cup \uparrow P && \text{since } x < c < p, \text{ for all } p \in P,\\
&\subseteq \updownarrow b \cup \{r\} \cup \uparrow P && \text{since } x \in \sigma(b).
\end{aligned}$$

Hence for all $x \in \sigma(b) \cap R'$, $x \in \sigma(b)'$. Note that in this case, $r \notin \sigma(b)$.

Finally, we consider $\mathsf{deleteRole}_2$. The proof is exactly the same as for Theorem 10, although we note that, in general, $\mathsf{deleteRole}_2(a, r)$ may delete another administrator r, thereby destroying the nested administrative domain $\sigma(r) \subset \sigma(a)$. ■

Proof of Corollary 12 We consider the operation $\mathsf{addEdge}_3(a, c, p)$. Then $p \in \sigma(a)$ and we have $[p] \subseteq \sigma(a) = [c]$. Hence, by Theorem 11, $\mathsf{addEdge}_3(a, c, p)$ is 2SP. The proofs for the other operations are similar and are omitted. ■

Proof of Theorem 13 We consider the operation $\mathsf{addEdge}_3(a, c, p)$. Suppose, in order to obtain a contradiction, that there exists $b < a$ such that $\mathsf{addEdge}_3(b, c, p)$ succeeds. Then we have $c, p \in \sigma(b)$, which implies that $[c] \subseteq \sigma(b) \subset \sigma(a)$, which is the required contradiction (since $\mathsf{addEdge}_3(a, c, p)$ succeeds only if $[c] = \sigma(a)$). The proof for the other operations are similar and are omitted. ■