# Authorization and antichains

*by*

**Jason Crampton**

*A thesis submitted in fulfilment of the requirements for the Degree of*

**Doctor of Philosophy**

*in the*
*University of London*

**February 2002**
**School of Computer Science and Information Systems**
**Birkbeck College**

# Abstract

Access control has been an important issue in military systems for many years and is becoming increasingly important in commercial systems. There are three important access control paradigms: the Bell-LaPadula model, the protection matrix model and the role-based access control model. Each of these models has its advantages and disadvantages. Partial orders play a significant part in the role-based access control model and are also important in defining the security lattice in the Bell-LaPadula model. The main goal of this thesis is to improve the understanding and specification of access control models through a rigorous mathematical approach.

We examine the mathematical foundations of the role-based access control model and conclude that antichains are a fundamental concept in the model. The analytical approach we adopt enables us to identify where improvements in the administration of role-based access control could be made. We then develop a new administrative model for role-based access control based on a novel, mathematical interpretation of encapsulated ranges. We show that this model supports discretionary access control features which have hitherto been difficult to incorporate into role-based access control frameworks.

Separation of duty is an important feature of role-based access control models that has usually been expressed in first-order logic. We present an alternative formalism for separation of duty policies based on antichains in a powerset (Sperner families), and show that it is no less expressive than existing approaches. The simplicity of the formalism enables us to analyze the complexity of implementing separation of duty policies. In the course of this analysis we establish new results about Sperner families.

We also define two orderings on the set of antichains in a partially ordered set and prove that in both cases the resulting structure is a distributive lattice. This lattice provides the formal framework for a family of secure access control models which incorporate the advantages of existing paradigms without introducing many of their respective disadvantages. We present two members of this family: a new model for role-based access control, for which we give an operational semantics and prove a security theorem similar to the Basic Security Theorem for the Bell-LaPadula model; and the secure hierarchical protection matrix model which combines the strong security properties of the Bell-LaPadula model with the flexibility of the protection matrix model.

To my mother

*I know no safe depository of the ultimate powers of the society but the people themselves; and if we think them not enlightened enough to exercise their control with a wholesome discretion, the remedy is not to take it from them, but to inform their discretion by education.*

Thomas Jefferson

# Contents

# List of Figures

# List of Tables

# List of Commands

# Acknowledgements

I must first thank my supervisor, Professor George Loizou, for giving me the opportunity to study for a PhD. It was a "leap of faith" on his part for which I am extremely grateful: the last three years have been very rewarding both intellectually and personally. His experience, technical ability, patience, generosity and attention to detail have made a huge contribution to the quality and accuracy of my work.

The quality of the final draft of this thesis has been significantly improved by the input of many people: Dr Szabolcs Mikulás made several perceptive comments on early drafts of the mathematical material; Steve Counsell, Conrad Crampton, Barbara Crampton, Uli Grohlke, Roger Mitton, James Skene, Steve Swift and Alan Tucker proof read and commented on sections of the thesis; Dr Roger Mitton has provided excellent practical advice on matters of style and grammar; Ian Lawson has suggested several improvements to the material on role-based access control; finally, I would particularly like to thank Anne Yarr for her meticulous proof reading of the first draft of the thesis and for her assistance with indexing and references.

I am grateful for the support and genuine, if somewhat apprehensive, interest of my family and friends. Particular thanks go to Anna-Maria, Annie, Carole, Hal, Julie, Paul, Sarah, Szabolcs, Tim, my mother, and especially my brother, Conrad. My attempts to explain my work to them can only have improved the clarity of my arguments. I would also like to thank Dr Trevor Fenner and James Skene for the interest and enthusiasm they have shown in my work, and for the many stimulating conversations we have had.

I would like to thank Dr Greg O'Shea for the time he spent discussing his work with me despite his busy schedule, and for the benefit of his experience and expertise in access control.

I would also like to mention the courtesy and assistance extended to me by the academic community – nearly all my enquiries have received a prompt and constructive response. I am particularly grateful to Dr Brian Davey, Dr Len LaPadula and Professor Ravi Sandhu for their assistance.

Finally, I would like to thank all the academic and support staff at Birkbeck College for their good-humoured and able assistance.

# Chapter 1

# Introduction

The security of the information in multi-user computer systems has been an important consideration for many years, and has become increasingly significant with the evolution of interconnected and distributed systems. Anderson (1973) identified three categories of security violations: unauthorized information release, unauthorized information modification and unauthorized denial of use. More recently, Stoneburger (2000) at the National Institute for Standards and Technology (NIST) suggests that information system security goals include *confidentiality*, *integrity*, *availability*, *accountability* and *assurance*, the first three of which clearly correspond to the prevention of the security violations identified by Anderson nearly thirty years earlier.[1]

An *information system security architecture* contains three key components: *authentication*, *audit* and *access control* (Joshi et al. 2001). In this thesis we focus on access control and how it can be used to realize the goals outlined above. In particular, it can help enforce confidentiality and data integrity, and can also facilitate assurance testing. Saltzer and Schroeder (1975) warn that access control should not be considered in isolation nor can it prevent all unauthorized use of information: rather, access control is but one aspect, albeit important, of a larger security strategy. However, we are justified in studying access control because system security ultimately depends on, and can be no stronger than, the underlying *access control mechanisms* (Stoneburger 2000).

It is necessary to make certain assumptions when considering access control. Firstly, to render the problem non-trivial, we assume that it is not the case that every subject can access every object (McLean 1990). Secondly, to ensure the problem is not meaningless we assume that authorized users act in good faith.[2] In short, we are concerned with ensuring that access to objects by valid, authorized subjects does not compromise the *access control policy* of the enterprise.

The main goal of this thesis is to extend existing access control models in order to provide a theoretical framework for access control mechanisms that can realize the goals of confidentiality and integrity. We demonstrate that by emphasizing the mathematical characteristics of access control we can elucidate the disadvantages of existing models, and provide the theoretical foundation for improvements to existing models and the development of new models.

[1]A glossary of common terms that are used without definition in this introduction can be found on page 188.
[2]This is analogous to assuming in the context of authentication, that users keep their passwords secret.

## 1.1 Access control

We now consider access control in the context of two paper-based systems. This enables us to draw some analogies with fundamental concepts in access control in computer systems. We will use this technique on several occasions in this thesis in order to introduce other aspects of access control.

In the first case, we assume that files are stored in filing cabinets under lock and key. Any user with an appropriate key can access the files. In this case, the access control mechanism is the filing cabinets and associated keys.

This form of access control mechanism is based on *capabilities*. That is, the user presents credentials (in our example, these credentials are represented by the keys to the filing cabinets) to the system that enable the system to determine whether access should be permitted or denied. In a computer system a *capability list* for a subject is generally considered to be a list of pairs $(o, a)$ representing an object identifier $o$ and a means of access $a$, such as "read" for example.

In the second case, we assume that files are released to users by a security guard who maintains lists of people who are authorized to read the files. If a user wishes to read a file, the security guard checks whether that user's name is in the list of people who are authorized to read that file. If it is, then the file is released. In this case, the access control mechanism is the security guard and the lists of authorized users.

This form of access control mechanism is based on *access control lists*. That is, each object in the system is associated with a list of users who are entitled to access that object. A request from a user to access a given object is only granted if the user is in the access control list. In a computer system an *access control list* for an object is a list of pairs $(s, a)$ where $s$ is a subject and $a$ is a means of access.

These two examples also illustrate the common assumption that access control should be based on permission not exclusion. Saltzer and Schroeder (1975) refer to this as *fail-safe defaults*. In other words, the default situation is lack of access and the access control mechanism explicitly identifies conditions under which access is permitted.

Controlling changes to the authorizations available in an access control mechanism is a fundamental problem in access control. We first consider how changes can be made to authorizations in our paper-based system. We assume that each user owns (or is responsible for) a set of files. In the first system, we imagine that each user keeps his files in a single filing cabinet, and to grant access to those files, the owner gives another user a key for the cabinet. In the second system, the owner amends the list of users who can access his files. This situation is known as *discretionary access control* in which users are responsible for the propagation of access rights. Discretionary access control is the most widely used form of access control. Operating systems such as UNIX and Windows NT implement discretionary access control using access control lists. For example, given the following output from the UNIX `ls` command,

```
-rwxr--r--   1 jason  research    1458 Sep  7 21:29 thesis.tex
```

the owner of the file (`jason`) would be granted `r`ead, `w`rite or e`x`ecute access to the file `thesis.tex`, while users in the group `research` would be granted read access but denied other forms of access (as would any other user of the system).[3] Furthermore, `jason`, as the owner of the file, can grant or revoke access rights using the `chmod` command.

There is substantial evidence that discretionary access control mechanisms are rarely configured correctly and contain many potential security violations (O'Shea 1997). The most significant causes of such situations are the huge complexity of maintaining hundreds of thousands of access control lists with little assistance from software tools and the lack of a clearly articulated security policy against which the performance of the access control mechanism can be measured.

> *" ... existing models, using capabilities or access control lists, leave the security administrator with an impossibly large task."* Moffett and Sloman (1991)

> *"If privilege components ... and protection components ... are to be specified by enumerations, then both the assignment of privileges and the testing ... will be too difficult to be of practical use."* Rabin and Tygar (1987)

However, it may be the case that users are not the owners of information, and are thus not entitled to make decisions about the propagation of access rights. A simple example concerns medical records, where doctors and health workers are responsible for maintaining the information, but are not entitled, in general, to release that information to third parties. This situation is known as *mandatory access control* in which the propagation of access rights is determined by some higher authority, which may be imposed by statutory or enterprise requirements.

The best known mandatory access control model is the Bell-LaPadula model. We explain the implementation of the Bell-LaPadula model using an analogy with the paper-based filing cabinet system. We imagine that the filing cabinets are distributed among several nested rooms each with a lock on the door, and that each user is given a door key. The key to the innermost room can open the door to every other room. The key to the outermost room can only open the door to the outermost room, etc.

Hence, if a user is given access to a file (that is, given a key to the filing cabinet), that access is only useful if the user is also able to open the door to the room containing the filing cabinet. Similarly, if a user is given a key to a room, that key is only useful if the user also has a key for a filing cabinet in that room. Computer systems which implement this kind of access control are often called *multi-level secure systems.* This additional restriction on the use of access rights is referred to as an *information flow policy.* Multi-level secure systems are commonly used in military applications and commercially sensitive applications. For example, the rooms could correspond to the US military security levels `unclassified`, `classified`, `secret` and `top secret`.

In short, although the propagation of access rights is not necessarily constrained in a multi-level secure system, additional checks implemented by the reference monitor prevent the use of authorizations that are not sanctioned by the information flow policy. Unfortunately, multi-level secure systems have been found to be too restrictive for general purpose use.

Of course many commercial applications will have access control requirements or *access control policy.* For example, it may be a requirement that personnel staff cannot view financial information and that no finance staff (with the exception of payroll staff, say) can view personnel information.

[3]Strictly speaking, a subject is a program or process being run by a user. However, where convenient we adopt the common practice of regarding a user as a subject.

The inflexibility of multi-level secure systems has meant that access control policies are usually implemented on discretionary access control mechanisms which, as we observed earlier, are known to be difficult to configure correctly. The problem is compounded by the fact that there is rarely an easy mapping between access control policy statements and access control mechanism primitives.

There have been several attempts to develop formal tools for specifying access control policies (Abadi et al. 1993; Heydon et al. 1989; Jajodia et al. 1997; Woo and Lam 1993). We do not consider access control policies in general in this thesis, although in Chapter 7 we consider *separation of duty* policies which have received considerable attention in the role-based access control community (Ahn and Sandhu 2000; Gligor et al. 1998) and for certain military applications (McLean 1990). Important contributions to the understanding of separation of duty in role-based access control have been made by Ahn and Sandhu (2000), Gavrila and Barkley (1998) and Gligor et al. (1998). These contributions have used first-order logic to describe a variety of separation of duty policies, but little work has been done on the difficulty of implementing and combining such policies.

Role-based access control models (Ferraiolo et al. 1999; Nyanchama and Osborn 1999; Sandhu et al. 1996) are a relatively new development. The motivation for such models is to ameliorate the problems of administration and flexibility identified with discretionary and mandatory access control, respectively. The fundamental concepts of role-based access control are now well established and are detailed in the recent unified standard for role-based access control (Ferraiolo et al. 2001). The basic idea is to reduce the complexity of access control administration by associating users and capabilities with roles. In our filing cabinet system this is analogous to reducing the number of keys in the system by issuing users with skeleton keys that can open many different filing cabinets.

However, there is less agreement on how role-based access control mechanisms should be administered. The most significant attempt in this area is ARBAC97 (Sandhu et al. 1999) which subsumes ideas developed by Nyanchama and Osborn (1999) and Gavrila and Barkley (1998). Although ARBAC97 is an important contribution to understanding and modelling administration in role-based access control, we believe that there are still substantial opportunities for research into administration in role-based access control. In addition, we believe that certain assumptions that have been made in the development of the RBAC96 and ARBAC97 models have compromised the original motivation and benefits of role-based access control.

Informally, the *safety problem* (Budd 1983; Harrison et al. 1976; Harrison and Ruzzo 1978; Lipton and Snyder 1977; Lipton and Snyder 1978; Sandhu 1992d; Sandhu 1992c) asks whether it is possible for a subject to acquire a certain access right to an object (Harrison et al. 1976). The safety problem in general is intractable for even relatively simple access control models; for more complex models – the protection matrix model, for example – it is undecidable. A model that has an undecidable safety problem is described as having *weak security properties*. The quest for access control models that provide sufficient flexibility for real-world applications and have strong security properties has occupied many researchers in the last 30 years (Bell and LaPadula 1973a; Budd 1983; Harrison et al. 1976; Harrison and Ruzzo 1978; Lipton and Snyder 1977; Lipton and Snyder 1978; Sandhu 1992c; Sandhu 1992d). However, we are unaware of any attempt to assess the complexity of the safety problem in role-based access control.

## 1.2 Partial orders

The second main strand in this thesis concerns the study of partial orders. Everyone has an intuitive idea about the properties of an ordering on a set. The most obvious thing about a statement about order is that it requires a comparison of two members of the set under consideration. We cannot say "bill is older than", for example. In other words, we can view an ordering as a *binary relation* on a set.

From the statements "bill is older than john" and "john is older than dave" we deduce that bill is older than dave. That is, order is a *transitive* property. Similarly we would not deduce that john is older than bill. That is, order is an *anti-symmetric* property.

The "is older than" relation is a *total ordering* on the set of people. That is, given two people an ordering always exists between them with respect to their ages. However this is not always the case. Consider the set of non-zero positive integers, and say "$x$ is less than $y$" if $x$ divides $y$ without remainder, which we will denote $x \mid y$. For example, $2 \mid 4$ and $8 \mid 72$. However, $3 \nmid 7$ and $7 \nmid 3$. We say that $\{3, 7\}$ is an *antichain*, and that $\mid$ is a *partial ordering*. In general, any set whose elements are pairwise incomparable (with respect to the ordering) is an antichain. For example, the set of primes is an infinite antichain in the set of non-zero natural numbers with the $\mid$ ordering. (Note that $x \mid x$ for all positive integers and hence $\mid$ is a *reflexive* binary relation.) We define partially ordered sets and antichains formally in Chapter 2.

Davey and Priestley (1990) quote several interesting applications of partially ordered sets: Boolean algebra and topology in mathematics; domain theory in computer science; and concept theory and analysis in the social sciences. In the context of access control, partially ordered sets are used to model information security policies (Denning 1976) and role hierarchies (Sandhu et al. 1996), and also arise in the typed access matrix model (Sandhu 1992c) and the schematic protection model (Sandhu 1988). In addition, Harrison and Ruzzo (1978) define a partial order on protection matrices in their analysis of the safety problem in mono-conditional monotonic protection systems.

In Chapters 3, 4 and 7 we will demonstrate that antichains are fundamental to the modelling of role-based access control and separation of duty policies. In Chapter 5 we prove, by constructing antichains in the role hierarchy, that the safety problem in role-based access control is undecidable. Finally, in Chapter 8, we show that sets of antichains can be used to provide a theoretical basis for a new framework for access control.

## 1.3 Outline of the thesis

In Chapter 2 we introduce some prerequisite concepts in mathematics and access control. In Section 2.1 we formally define partially ordered sets and associated standard results and definitions. Of particular importance in this section are the definitions of a *lattice* and of a *completion* of a partially ordered set. In Section 2.2 we introduce the combinatorial concepts that will be required in Chapter 7 when we consider the structural complexity of separation of duty policies. In Sections 2.3.1 and 2.3.2 we describe the protection matrix model and the Bell-LaPadula model, respectively. The protection matrix (Lampson 1971; Harrison et al. 1976) is a conceptual tool for modelling discretionary access control mechanisms. Access control lists and capabilities can be expressed in a natural way in this model. We conclude the chapter with a brief outline of complexity theory and deterministic Turing machines which we use when discussing the safety problem in Chapter 5.

Chapter 3 is a comprehensive review of RBAC96 and ARBAC97. It introduces a mathematical notation for role-based access control based on ideas introduced in Section 2.1, which we believe is far easier to use than existing schemes. We also analyze the complexity of certain administrative functions required by the ARBAC97 model. This leads naturally to the material in Chapter 4 in which we discuss the shortcomings of existing role-based access control models.

The simplicity of our notation and its mathematical basis facilitates a rigorous analysis of the RBAC96 and ARBAC97 models. This analysis suggests certain new approaches to administration in role-based access control. In Chapter 4, we introduce RHA, a new family of models for administration of the role hierarchy and extend this to the SARBAC model, a complete model for administration in role-based access control. The development of RHA is based on intuitive assumptions about the way in which administration should be performed; as a result, we believe that RHA achieves a more realistic and flexible solution to administration in a role-based access control environment than ARBAC97. In addition, we show that discretionary access control can be supported using SARBAC. Previous attempts to incorporate discretionary access control into role-based access control have proved rather complicated and counter-intuitive (Osborn et al. 2000; Sandhu and Munawer 1998a). We conclude the chapter with a description of our generalized role-based access control model which extends the concepts of *group* and *ability* as used in Sandhu et al. (1999).

Chapter 5 is concerned with the safety problem. There have been many results published on the safety problem in the protection matrix model and its variants in the last 25 years (Harrison et al. 1976; Harrison and Ruzzo 1978; Lipton and Snyder 1978; Sandhu 1992c). We review these results and sketch some of the methods that have been employed to analyze the safety problem. The main contribution of Chapter 5 is to define the safety problem in role-based access control and establish that in general it is undecidable in the RBAC96/ARBAC97 framework (Crampton and Loizou 2001c). We also prove that the safety problem is undecidable in the context of RBAC96 using SARBAC as the administrative model.

In Chapter 6, we prove that the set of all antichains in a finite partially ordered set $X$ forms a lattice under two different orderings, and that this lattice is a completion of $X$. We also determine the binary operations of the two lattices, prove analogues of Birkhoff's Representation Theorem for finite distributive lattices and exhibit a Dedekind-MacNeille-style completion of $X$. These results appeared in Crampton and Loizou (2000) and Crampton and Loizou (2001b). One ordering on the lattice of antichains is used in Chapter 7 to support binary operations on conflict of interest policies. The second ordering provides the formal motivation for a family of secure access control models which we develop in Chapter 8.

In Chapter 7, we formally define a very simple model for separation of duty policies based on antichains in the powerset of a finite set and compare it to existing approaches. The inherent simplicity of our representation of these policies and their formal equivalence to Sperner families enables us to establish several important results on the structural complexity of a separation of duty policy and on the number of separation of duty policies. We show that a binary operation can be defined on separation of duty policies that captures the idea of composition of two policies in a natural and intuitive way. In the course of this chapter we also identify some novel applications of separation of duty policies in role-based access control.

In Chapter 8 we introduce a framework for the development of secure access control models. This framework is based on ideas in the Bell-LaPadula model, role-based access control and the lattice of antichains defined in Chapter 6. The main contribution of this chapter is to develop a new model for role-based access control from this framework which uses parts of the administrative model developed in Chapter 4. A preliminary version of this model and the work on separation of duty policies appeared in (Crampton and Loizou 2001a). The second contribution of this chapter is to develop the secure hierarchical protection matrix model which reflects commercial enterprises more accurately than the Bell-LaPadula model while preserving its strong security properties.

Finally, in Chapter 9, we review the contributions of the thesis and discuss the numerous opportunities for future research.

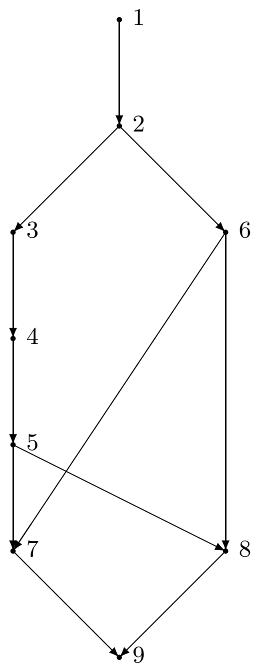
The contributions of this thesis are summarized in Table 1.1. Figure 1.1 is a flow chart that summarizes the contents of the thesis and the order in which the chapters can be read. Figures, tables, commands and equations are numbered sequentially within each of the nine chapters. Theorems, definitions and similar environments are numbered sequentially within each section. Citations follow the recommendations of The Chicago Manual of Style using the name of the author(s) and the year of publication. The full details of the references are given at the end of the thesis in alphabetical order by author.

| | |
|---|---|
| Section 3.2 | Notation for role-based access control |
| Section 3.6 | Complexity of administrative functions in ARBAC97 |
| Section 4.2 | New model for administration of role hierarchy |
| Section 4.3 | New model for administration in role-based access control |
| Section 4.4 | Model to support discretionary access control features |
| Section 4.5 | The generalized role hierarchy |
| Section 5.2 | Undecidability of the safety problem in role-based access control |
| Chapter 6 | An analysis of the lattice of antichains in a partially ordered set |
| Section 7.1 | A new approach to separation of duty policies |
| Section 7.2.2 | Novel applications of separation of duty policies in role-based access control |
| Section 7.3 | New bounds on the number of Sperner families |
| Chapter 8 | A new framework for secure access control models |
| Section 8.1 | A secure role-based access control model |
| Section 8.2 | A secure protection matrix model |

Table 1.1: Contributions of the thesis

Finally, we briefly describe the symbolic conventions employed in the thesis. In general we will denote functions by lower case Greek letters; elements of a set in lower case letters ($x, y \in X$, for example); subsets of a set in upper case letters ($X_1, X_2 \subseteq X$, for example); families of sets in upper case script letters ($\mathcal{F}$, for example). Clearly, there will be instances where these conventions cannot be adhered to. For example, an element of $\mathcal{F}$ could be denoted as a lower case script letter or as an upper case roman letter since it is a set. In general we will adopt the latter course.

We believe it is always desirable to choose notation and symbols that assist the reader. However, this means that certain symbols may be used more than once. The reader should bear in mind therefore that the semantics of a symbol may change, although this occurs rarely and never



| | |
|---|---|
| 1 | Introduction |
| 2 | Preliminaries |
| 3 | Role-based Access Control |
| 4 | Extensions to Role-based Access Control |
| 5 | The Safety Problem |
| 6 | Completions of a Poset |
| 7 | Conflict of Interest Policies |
| 8 | The Secure Hierarchical Authorization Framework |
| 9 | Conclusions and Future Work |

**Figure 1.1:** Structure of the thesis

within the scope of a section. We trust that the reader finds that this hint of ambiguity in the notation is more than offset by the use of intuitive and semantically rich symbolism. A list of notation used in the thesis can be found on page 185.

# Chapter 2

# Preliminaries

The purpose of this chapter is to acquaint the reader with the prerequisite mathematical material.

Section 2.1 introduces elementary definitions and results for partially ordered sets (posets). In particular, we introduce the notions of an *antichain* and a *completion* which are fundamental to the material in Chapter 6. Further details can be found in any book on order and lattice theory, of which the following are recommended to the interested reader: Birkhoff (1948); Burris and Sankappanavar (1981); Davey and Priestley (1990) and Grätzer (1978).

Section 2.2 introduces the combinatorial material that will be used extensively in Chapter 7. Central to this section is the concept of a *symmetric chain partition* which provides a highly structured method of partitioning the powerset of a finite set. An antichain in the powerset of a finite set is often referred to as a *Sperner family*, and is formally equivalent to a canonical conflict of interest policy which is defined in Chapter 7. Symmetric chain partitions can be used to prove Sperner's Theorem which imposes a maximal size on a Sperner family. We state and prove a result due to Hansel which imposes upper and lower bounds on the number of Sperner families in a given powerset (Hansel 1966). This prepares the reader for the proof of Theorem 7.3.2. An excellent introduction to symmetric chain partitions and Sperner's Theorem can be found in Brualdi (1999). A detailed account of Sperner families and Hansel's result can be found in Engel (1997).

In Section 2.3 we consider the two best known access control models, the protection matrix model and the Bell-LaPadula model. This material serves as a useful introduction to access control models and informs the material in Chapters 5 and 8.

We conclude the chapter with short, fairly informal introductions to complexity theory and Turing machines. This material will be required in the discussion of the safety problem in Chapter 5.

## 2.1 Partial orders

**Definition 2.1.1** *A pair* $\langle X, \leqslant \rangle$ *is a* partially ordered set *or* poset *if for all* $x, y, z \in X$,

- $x \leqslant x$,
- $x \leqslant y$ *and* $y \leqslant x$ *implies* $x = y$,
- $x \leqslant y$ *and* $y \leqslant z$ *implies* $x \leqslant z$.

In other words, $\leqslant$ is a binary relation on $X$ that is reflexive, anti-symmetric and transitive. We refer to $\leqslant$ as a *partial order*. Henceforth, when the ordering is clear from context, we will write $X$ to mean the poset $\langle X, \leqslant \rangle$. We will write: $x < y$ if $x \leqslant y$ and $x \neq y$; $x \geqslant y$ and $y \leqslant x$ interchangeably; $x \parallel y$ if $x \nleqslant y$ and $y \ngeqslant y$. A reflexive, transitive binary relation $\leqslant$ on $X$ is called a *pre-order*.

Unless otherwise stated the set $X$ is finite and has cardinality $n$. The *powerset* of $X$ is the set of subsets of $X$, and is denoted $2^X$. We will identify $X = \{x_1, \ldots, x_n\}$ with the set $[n] = \{1, \ldots, n\}$ under the mapping $x_i \mapsto i$. It can be easily verified that the following are all examples of posets:

- $\langle 2^X, \subseteq \rangle$;
- $\langle \mathbb{N}, \leqslant \rangle$, where $\leqslant$ is the standard ordering on the set of natural numbers $\mathbb{N}$;
- $\langle \mathbb{N}^+, | \rangle$, where $\mathbb{N}^+$ is the set of non-zero natural numbers and $m \mid n$ if $m$ divides $n$ without remainder.

**Definition 2.1.2** *Let $X$ be a poset and $x, y \in X$. 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 $x = z$.*

The poset $\langle 2^{[n]}, \subseteq \rangle$ plays an important part in this thesis. Therefore, for notational convenience we will freely interchange the order symbols $\subseteq$ and $\leqslant$. In particular, for $Y, Z \subseteq X$, $Y \lessdot Z$ means $Y \subset Z$ and $|Y| = |Z| - 1$, where $|Y|$ is the cardinality of $Y$. For example $\{1\} \lessdot \{1, 2\}$, but $\{1\} \not\lessdot \{1, 2, 3\}$ and $\{1\} \not\lessdot \{2, 3\}$. Hence the reader should be careful to distinguish between the use of $\leqslant$ (and $<$) for subsets and integers. This distinction will always be clear from context.

A poset $\langle X, \leqslant \rangle$ is often represented by a *Hasse diagram*, which is the graph of the *covering relation* on $X$. That is, the nodes of the graph are members of $X$, and an edge exists between $x$ and $y$ if $x \lessdot y$. In particular, an edge is not drawn from a node to itself, nor is an edge drawn between $x$ and $y$ if there exists $z$ such that $x < z < y$.

By convention, if $x \lessdot y$, the node labelled $x$ will be lower than the node labelled $y$ in the Hasse diagram. We will also adopt the convention that nodes of interest will be represented by circles (see Figure 2.1a, for example), and the remaining nodes will be represented by discs.[1] Three posets, represented by their Hasse diagrams, are shown in Figure 2.1.

**Definition 2.1.3** *Given $x \in X$, $\nabla x = \{y \in X : x \lessdot y\}$ is defined to be the* upper shadow *of $x$; $\Delta x = \{y \in X : y \lessdot x\}$ is the* lower shadow *of $x$.*

**Definition 2.1.4** *If $X$ is a poset, $Y \subseteq X$ is a* chain *if, for all $y_1, y_2 \in Y$, either $y_1 \leqslant y_2$ or $y_2 \leqslant y_1$. $Y$ is an* antichain *if, for all $y_1, y_2 \in Y$, either $y_1 = y_2$ or $y_1 \parallel y_2$.*

We denote the set of antichains in a poset $X$ by $\mathcal{A}(X)$. The antichain $\{b, c, d\}$ is highlighted in Figure 2.1a. An antichain in $2^{[n]}$ is also referred to as a *Sperner family* (Engel 1997). That is, a Sperner family is a collection of subsets of $[n]$ in which no member of the collection is a subset of any other. For example, $\{\{1\}, \{2, 3\}, \{2, 4, 5\}\}$ is a Sperner family in $2^{[n]}$ for $n \geqslant 5$.

[1]In this context, a disc is a circle and its interior.

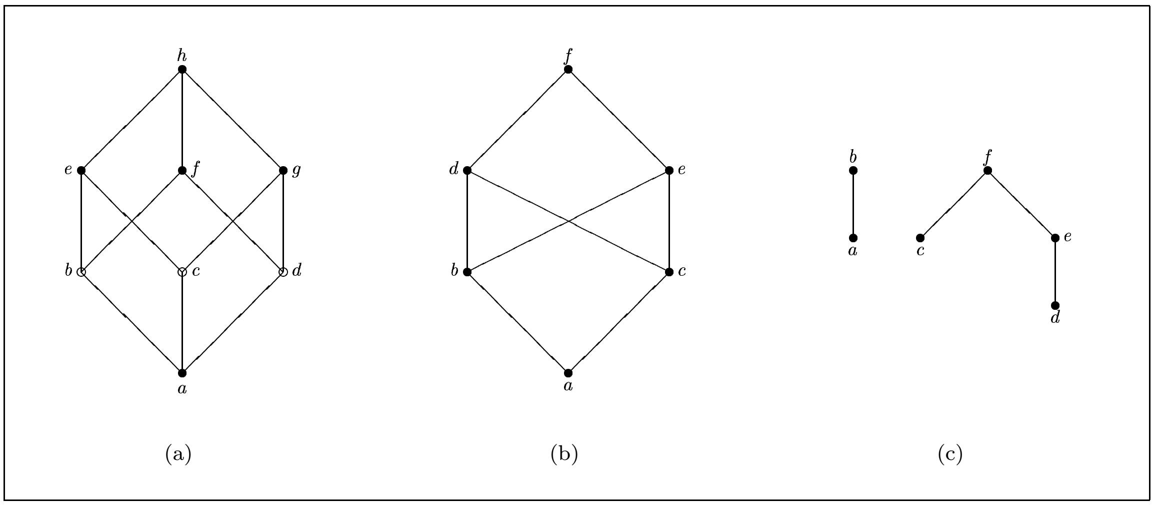


**Figure 2.1:** Hasse diagrams

**Definition 2.1.5** *The* width *of a poset, $X$, denoted $w(X)$, is the maximal cardinality of an antichain in $X$.*

**Definition 2.1.6** *Given a poset $X$ and $Y \subseteq X$, we say $y \in Y$ is a* minimal *element if for all $y' \in Y$, $y' \leqslant y$ implies $y = y'$. Similarly, $y \in Y$ is a* maximal *element if for all $y' \in Y$, $y \leqslant y'$ implies $y = y'$.*

**Definition 2.1.7** *Let $X$ be a poset, and let $Y \subseteq X$.*

- *An element $x \in X$ is an* upper bound *for $Y$ if, for all $y \in Y$, $y \leqslant x$. We denote the set of upper bounds of $Y$ by $Y^u$.*
- *If $Y^u$ has a unique minimal element $x$, then we say $x$ is the* least upper bound *or* supremum *of $Y$. We denote the supremum of $Y$ by* $\sup Y$.
- *An element $x \in X$ is a* lower bound *for $Y$ if, for all $y \in Y$, $x \leqslant y$. We denote the set of lower bounds of $Y$ by $Y^l$.*
- *If $Y^l$ has a unique maximal element $x$, then we say $x$ is the* greatest lower bound *or* infimum *of $Y$. We denote the infimum of $Y$ by* $\inf Y$.

**Definition 2.1.8** *A poset $X$ is a* lattice *if, and only if, for all $x, y \in X$ both* $\inf\{x, y\}$ *and* $\sup\{x, y\}$ *exist in $X$. If for all $Y \subseteq X$,* $\sup Y$ *and* $\inf Y$ *exist (in $X$), then $X$ is called a* complete *lattice.*

It can be easily checked that Figure 2.1a represents a lattice. However, the poset represented in Figure 2.1b is not a lattice since $\{b, c\}$ does not have a least upper bound. (Similarly, $\{d, e\}$ does not have a greatest lower bound.) The poset in Figure 2.1c is not a lattice because, for example, $\{b, f\}$ does not have an upper bound.

Clearly all finite lattices are complete. (Let $X$ be a lattice. Then by definition, for all $Y \subseteq X$ such that $|Y| = 2$, $\sup Y$ and $\inf Y$ exist. By induction, for all $Y \subseteq X$ such that $Y$ is finite, $\sup Y$ and $\inf Y$ exist. Hence, if $X$ is finite, every subset of $X$ has a greatest lower bound and a least upper bound.) However, note that $\langle \mathbb{N}, \leqslant \rangle$ is a lattice (since for all $m, n \in \mathbb{N}$ such that $m < n$, $\inf\{m, n\} = m$ and $\sup\{m, n\} = n$) but is not a complete lattice because, for example, $\{x \in \mathbb{N} : x > 2\}$ does not have a least upper bound in $\mathbb{N}$.

If $L$ is a lattice $\inf\{x, y\}$ is usually written $x \wedge y$ (the "meet" of $x$ and $y$) and $\sup\{x, y\}$ is usually written $x \vee y$ (the "join" of $x$ and $y$). We will also write $\langle L, \vee, \wedge \rangle$ to mean that the set $L$ is a lattice with the operations $\vee$ and $\wedge$. For example, $\langle 2^{[n]}, \cup, \cap \rangle$ is a lattice. Indeed a lattice can be defined as a purely algebraic structure in terms of these operations.

**Definition 2.1.9** *Let $\langle X_1, \leqslant_1 \rangle$ and $\langle X_2, \leqslant_2 \rangle$ be two posets. Then $\phi : X_1 \to X_2$ is*

- *an* order-preserving function *if $x \leqslant_1 y$ implies $\phi(x) \leqslant_2 \phi(y)$,*
- *an* order embedding *if $x \leqslant_1 y$ if, and only if, $\phi(x) \leqslant_2 \phi(y)$.*

*If $\phi$ is an order embedding we will write $\phi : X_1 \hookrightarrow X_2$.*

**Definition 2.1.10** *Let $X$ be a poset. If $\phi : X \hookrightarrow L$, where $L$ is a complete lattice, then we say that $L$ is a* completion *of $X$.*

**Definition 2.1.11** *Two lattices, $\langle L_1, \vee_1, \wedge_1 \rangle$, $\langle L_2, \vee_2, \wedge_2 \rangle$, are* isomorphic*, denoted $L_1 \cong L_2$, if there is a bijection $\phi : L_1 \to L_2$ such that $\phi(a \vee_1 b) = \phi(a) \vee_2 \phi(b)$ and $\phi(a \wedge_1 b) = \phi(a) \wedge_2 \phi(b)$ for all $a, b \in L_1$.*

Informally, two lattices are isomorphic if their Hasse diagrams have the same structure. For example, the lattice in Figure 2.1a is isomorphic to $\langle 2^{[3]}, \subseteq \rangle$.

The following result characterizes isomorphic lattices using the respective partial orderings on the lattices. It will be used in the proofs of Theorem 6.1.1 and Theorem 6.2.1.

**Theorem 2.1.1** *Two lattices $\langle L_1, \leqslant_1 \rangle$ and $\langle L_2, \leqslant_2 \rangle$ are isomorphic if, and only if, there is a bijection $\phi$ from $L_1$ to $L_2$ such that both $\phi$ and $\phi^{-1}$ are order-preserving.*

**Definition 2.1.12** *If $X$ is a poset then $Y \subseteq X$ is an* (order) ideal *if for all $y \in Y$, $x \in X$, $x \leqslant y$ implies $x \in Y$. If $X$ is a poset then $Y \subseteq X$ is an* (order) filter *if for all $y \in Y$, $x \in X$, $x \geqslant y$ implies $x \in Y$. The set of order ideals of $X$ is denoted $\mathcal{I}(X)$. The set of order filters of $X$ is denoted $\mathcal{F}(X)$.*

**Definition 2.1.13** *If $X$ is a poset and $Y \subseteq X$, we define $\downarrow Y$ read "down $Y$" as follows:*

$$\downarrow Y = \{x \in X : \text{ there exists } y \in Y \text{ such that } x \leqslant y\}.$$

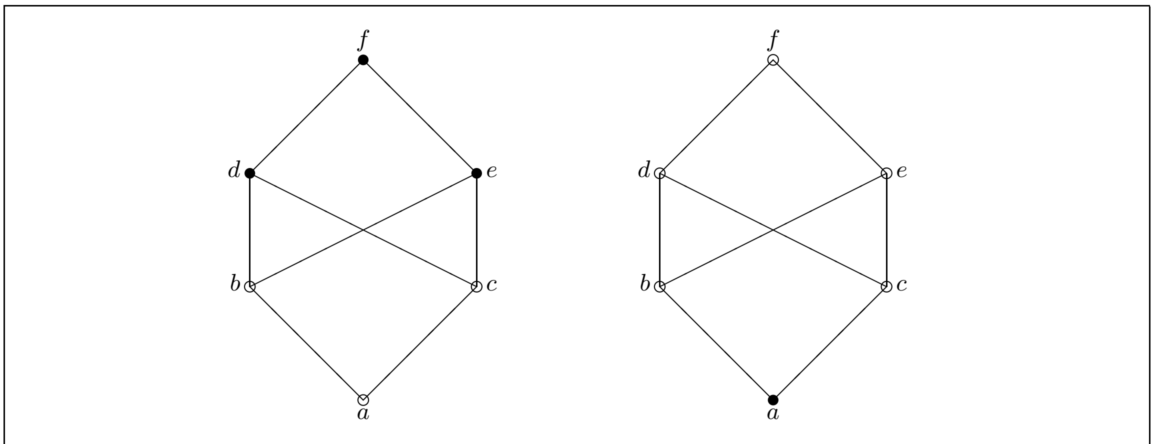
*Similarly we define $\uparrow Y$ read "up $Y$" as follows:*

$$\uparrow Y = \{x \in X : \text{ there exists } y \in Y \text{ such that } x \geqslant y\}.$$

*We will denote $\downarrow\{x\}$ by $\downarrow x$ and $\uparrow\{x\}$ by $\uparrow x$.*

**Proposition 2.1.1** *For all $Y \subseteq X$, $\downarrow Y \in \mathcal{I}(X)$, $Y \subseteq \downarrow Y$, $\uparrow Y \in \mathcal{F}(X)$ and $Y \subseteq \uparrow Y$.*

Figure 2.2 shows an order ideal and an order filter in the poset depicted in Figure 2.1b. It can be seen that the set of maximal elements in the ideal and the set of minimal elements in the filter both equal the antichain $\{b, c\}$. Furthermore, the ideal is $\downarrow \{b, c\}$ and the filter is $\uparrow \{b, c\}$. We will use this correspondence between ideals, filters and antichains extensively in Chapter 6.



(a) An ideal (b) A filter

**Figure 2.2:** Ideals and filters

**Lemma 2.1.1** *For any (possibly infinite) poset $X$, $\langle \mathcal{I}(X), \subseteq \rangle$ and $\langle \mathcal{F}(X), \supseteq \rangle$ are complete lattices. Furthermore, $\langle \mathcal{I}(X), \subseteq \rangle$ and $\langle \mathcal{F}(X), \supseteq \rangle$ are completions of $X$ via the mappings $x \mapsto \downarrow x$ and $x \mapsto \uparrow x$, respectively.*

The proof of Lemma 2.1.1 follows immediately from Definition 2.1.13 (Davey and Priestley 1990). Lemma 2.1.1 is used in conjunction with Theorem 2.1.1 in Chapter 6 to prove that the set of antichains is a complete lattice.

**Definition 2.1.14** *Given a poset $X$, a* range *is a subset of $X$ defined by two* end points, *$x, y \in X$. An* open *range, denoted $(x, y)$, is defined to be $\{z \in X : x < z < y\}$. A* closed *range, denoted $[x, y]$, is defined to be $\{z \in X : x \leqslant z \leqslant y\}$.*

Additionally, $(x, y] = \{z \in X : x < z \leqslant y\}$, and $[x, y) = \{z \in X : x \leqslant z < y\}$. Ranges are the basic unit of administration in ARBAC97, which is discussed in detail in Section 3.4.

**Definition 2.1.15** *Let $L$ be a lattice; $x, y, z \in L$ obey the* distributive law *if*

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$$

*If the distributive law holds for all $x, y, z \in L$ then $L$ is a* distributive *lattice.*

**Definition 2.1.16** *Let $L$ be a lattice. An element $x \in L$ is* join-irreducible *if $\Delta x = \{y\}$ for some $y \in L$. We denote the set of join-irreducible elements in $L$ by $\mathcal{J}(L)$.*

This is not the standard definition of a join-irreducible element, which is normally stated in terms of the join operation (Davey and Priestley 1990). However, it is a more intuitive definition and has an obvious interpretation in the context of a Hasse diagram.

**Lemma 2.1.2** *Let $L$ be a distributive lattice and let $x \in \mathcal{J}(L)$, where $x = a_1 \vee \cdots \vee a_k$ for $a_1, \ldots, a_k \in L$. Then $x \leqslant a_i$ for some $i$, $1 \leqslant i \leqslant k$.*

**Definition 2.1.17** *Given two posets $\langle X, \leqslant\rangle$ and $\langle Y, \leqslant\rangle$ we can form the following posets:*

- *The* dual *of $X$, denoted $X^{\partial}$, is the poset $\langle X, \geqslant\rangle$.*
- *The* linear sum *of $X$ and $Y$, denoted $X \oplus Y$, is the poset $\langle X \dot{\cup} Y, \leqslant\rangle$, where $X \dot{\cup} Y$ is the* disjoint union *of $X$ and $Y$ and $x \leqslant y$ in $X \oplus Y$ if, and only if,*

$$\begin{aligned} & x, y \in X \text{ and } x \leqslant y \text{ in } X, \\ \text{or } & x, y \in Y \text{ and } x \leqslant y \text{ in } Y, \\ \text{or } & x \in X,\ y \in Y. \end{aligned}$$

- *The* cartesian product *of $X$ and $Y$, denoted $X \times Y$, where $(x_1, y_1) \leqslant (x_2, y_2)$ in $X \times Y$ if, and only if, $x_1 \leqslant x_2$ and $y_1 \leqslant y_2$.*

The Hasse diagram of $X^{\partial}$ is obtained by inverting the Hasse diagram of $X$. For example, the dual of $\langle 2^{[n]}, \subseteq\rangle$ is $\langle 2^{[n]}, \supseteq\rangle$. The Hasse diagram of $X \oplus Y$ is obtained by placing the Hasse diagram of $Y$ above that of $X$ and inserting an edge between each maximal element of $X$ and each minimal element of $Y$. For example, the poset in Figure 2.1b is the linear sum of the posets $\{a, b, c\}$ and $\{d, e, f\}$. We will encounter the cartesian product of two posets in the Bell-LaPadula model in Section 2.3.2.

## 2.2 Combinatorial considerations

The main purpose of this section is to introduce *symmetric chain partitions* and a result due to Hansel (1966). These ideas yield several properties of antichains in a powerset that will be used extensively in Chapter 7. The ideas in this section are rather technical and are not required until Section 7.3. Therefore, the reader may prefer to omit this section now and return to it before reading Chapter 7.

**Definition 2.2.1** *A* partition *of a set $X$ is a collection of subsets of $X$, $\{X_1, \ldots, X_k\}$, such that $X = \bigcup_{i=1}^{k} X_i$ and $X_i \cap X_j = \emptyset$ for all $1 \leqslant i < j \leqslant k$.*

**Definition 2.2.2** *A* symmetric chain partition *of $2^{[n]}$ is a partition of $2^{[n]}$ into chains such that for each chain $\{C_0, \ldots, C_k\}$, $C_0 \lessdot C_1 \lessdot \ldots \lessdot C_k$ and $|C_0| + |C_k| = n$. The* length *of such a chain is defined to be $k + 1$.*

Note that a symmetric chain partition is not unique for $n \geqslant 2$. Two symmetric chain partitions of $2^{[3]}$ are shown in Figure 2.3.

**Theorem 2.2.1** *For all $n \geqslant 1$, there exists a symmetric chain partition of $2^{[n]}$.*

We present a constructive proof by induction of Theorem 2.2.1 (Brualdi 1999) as it introduces notation that will be used extensively in this section.

$$\begin{array}{ll}\emptyset \subset \{1\} \subset \{1,2\} \subset \{1,2,3\} \\ \qquad \{2\} \subset \{2,3\} \\ \qquad \{3\} \subset \{1,3\}\end{array} \qquad \begin{array}{ll}\emptyset \subset \{2\} \subset \{2,3\} \subset \{1,2,3\} \\ \qquad \{3\} \subset \{1,3\} \\ \qquad \{1\} \subset \{1,2\}\end{array}$$

(a) (b)

**Figure 2.3:** Symmetric chain partitions of $2^{[3]}$

**Proof of Theorem 2.2.1** We use the symmetric chain partition from Figure 2.3a as a base case for the induction.

Suppose now that there is a symmetric chain partition of $2^{[n]}$ for $3 \leqslant n \leqslant N$. For each chain $\mathcal{C} = C_0 \lessdot \ldots \lessdot C_k$, we construct the chains

$$\mathcal{C}^+ \triangleq C_0 \lessdot \ldots \lessdot C_k \lessdot C_k \cup \{N+1\} \quad \text{and} \tag{2.1}$$

$$\mathcal{C}^- \triangleq C_0 \cup \{N+1\} \lessdot \ldots \lessdot C_{k-1} \cup \{N+1\}. \tag{2.2}$$

(Note that $|\mathcal{C}^+| = |\mathcal{C}| + 1$ and $|\mathcal{C}^-| = |\mathcal{C}| - 1$. Therefore, if $|\mathcal{C}| = 1$ we only construct $\mathcal{C}^+$.) Clearly, by construction, the resulting chains form a symmetric chain partition of $2^{[n+1]}$. ■

Henceforth, for all $n \geqslant 3$, we will assume the existence of an inductively constructed symmetric chain partition, denoted $SCP_n$, using the symmetric chain partition in Figure 2.3a as the base case. The construction of $SCP_4$ from $SCP_3$ is shown in Figure 2.4.

$$\begin{array}{rl}\emptyset \subset \{1\} \subset \{1,2\} \subset \{1,2,3\} \to & \left\{\begin{array}{l}\emptyset \subset \{1\} \subset \{1,2\} \subset \{1,2,3\} \subset \{1,2,3,4\} \\ \{4\} \subset \{1,4\} \subset \{1,2,4\}\end{array}\right. \\ \{2\} \subset \{2,3\} \to & \left\{\begin{array}{l}\{2\} \subset \{2,3\} \subset \{2,3,4\} \\ \{2,4\}\end{array}\right. \\ \{3\} \subset \{1,3\} \to & \left\{\begin{array}{l}\{3\} \subset \{1,3\} \subset \{1,3,4\} \\ \{3,4\}\end{array}\right.\end{array}$$

**Figure 2.4:** An inductive construction of $SCP_4$

**Lemma 2.2.1** *$SCP_n$ has $\binom{n}{\lfloor n/2 \rfloor}$ chains.*

For a proof of this elementary result see Brualdi (1999), for example.

**Theorem 2.2.2 (Sperner 1928)** *For all $\mathcal{S} \in \mathcal{A}(2^{[n]})$,*

$$|\mathcal{S}| \leqslant \binom{n}{\lfloor n/2 \rfloor},$$

*with equality when*

$$\mathcal{S} = \begin{cases} \{S \subseteq [n] : |S| = \frac{n}{2}\} & n \text{ even}, \\ \{S \subseteq [n] : |S| = \frac{n-1}{2}\} \quad \text{or} \quad \{S \subseteq [n] : |S| = \frac{n+1}{2}\} & n \text{ odd}. \end{cases}$$

**Proof** (Sketch) Sperner's Theorem can be proved as a corollary of Lemma 2.2.1 by noting that if $\mathcal{S} \in \mathcal{A}(2^{[n]})$ then for all $\mathcal{C} \in SCP_n$, $|\mathcal{S} \cap \mathcal{C}| \leqslant 1$. $SCP_n$ is a partition of $2^{[n]}$; therefore, $|\mathcal{S}| \leqslant |SCP_n| = \binom{n}{\lfloor n/2 \rfloor}$. ■

In other words, Sperner's Theorem states the width of $2^{[n]}$. For example, the largest antichains in $2^{[3]}$ are $\{\{1\},\{2\},\{3\}\}$ and $\{\{1,2\},\{1,3\},\{2,3\}\}$, both of which have size $3 = \binom{3}{1}$, confirming the result of Sperner's Theorem. Demetrovics (1978) proved using Sperner's Theorem that the number of minimal keys in a database table with $n$ attributes is not greater than $\binom{n}{n/2}$, and this bound is the best possible.

Sperner's Theorem and its original proof (Sperner 1928) provide the inspiration for Lemma 7.3.1. The method of proof using symmetric chain partitions is also employed in Lemmas 7.3.2 and 7.3.3.

In fact, Sperner's Theorem can also be derived from the *LYM-inequality*, due independently to Lubell (1966), Yamamoto (1954) and Meshalkin (1963).

**Lemma 2.2.2 (LYM-inequality)** *Given a Sperner family* $\mathcal{S}$*,*

$$\sum_{S \in \mathcal{S}} \frac{1}{\binom{n}{|S|}} \leqslant 1.$$

**Remark 2.2.1** *Let* $n_k$ *be the number of elements of cardinality* $k$ *in* $\mathcal{S}$*. Then we can re-write the LYM-inequality as*

$$\sum_k \frac{n_k}{\binom{n}{k}} \leqslant 1.$$

**Theorem 2.2.3 (Hansel 1966)** *For all* $n \geqslant 1$*,*

$$2^\nu \leqslant |\mathcal{A}(2^{[n]})| \leqslant 3^\nu, \text{ where } \nu = \binom{n}{\lfloor n/2 \rfloor}.$$

To clarify the presentation of the proof of Theorem 2.2.3, we first prove two preparatory results: Proposition 2.2.1 and Theorem 2.2.4. We adopt the style of proof given in Engel (1997). We give a proof of Hansel's result because Theorem 7.3.2, in which we significantly improve on the upper bound in Theorem 2.2.3, is proved using a similar method.

**Proposition 2.2.1** *Let* $\mathcal{C} = C_0 \lessdot C_1 \lessdot \cdots \lessdot C_r$ *be a chain in* $SCP_n$*. If* $|C_0| = i$ *then* $|\mathcal{C}| = n-2i+1$*.*

**Proof** If $|C_0| = i$, then $|C_r| = n - i$, since $|C_0| + |C_r| = n$. Furthermore, $|C_0|, \ldots, |C_r|$ are consecutive integers, since $C_0 \lessdot \cdots \lessdot C_r$. There are $|C_r| - |C_0| + 1 = (n-i) - i + 1 = n - 2i + 1$ such integers. ■

**Theorem 2.2.4** *For every chain $\mathcal{C}$ in $SCP_n$, and for every set of three consecutive members $C_{i-1} \lessdot C_i \lessdot C_{i+1}$ of $\mathcal{C}$, there is some $C_i'$ such that $C_{i-1} \lessdot C_i' \lessdot C_{i+1}$ and $C_i'$ is contained in a chain $\mathcal{D}$ with $|\mathcal{D}| = |\mathcal{C}| - 2$.*

**Proof** (By induction on $n$) Clearly the theorem is true for the case $n = 3$ by inspection of Figure 2.3. Suppose the result is true for all $n \leqslant N$ and consider $SCP_{N+1}$. There are two cases to examine.

- $C_{i-1}, C_i, C_{i+1}$ belong to a chain of the form $\mathcal{C}^+ \in SCP_{N+1}$, defined in (2.1). There are two possibilities.
  - If $N+1 \notin C_{i+1}$, then $C_{i-1}, C_i, C_{i+1} \in \mathcal{C}$, where $\mathcal{C} \in SCP_N$, and by inductive hypothesis there exists a chain $\mathcal{D} \in SCP_N$ such that $C_i' \in \mathcal{D}$, $|\mathcal{D}| = |\mathcal{C}| - 2$ and $C_{i-1} \lessdot C_i' \lessdot C_{i+1}$. Furthermore, $C_i' \in \mathcal{D}^+$, $\mathcal{D}^+ \in SCP_{N+1}$ and $|\mathcal{D}^+| = |\mathcal{C}^+| - 2$, since $|\mathcal{D}| = |\mathcal{C}| - 2$.
  - If $N + 1 \in C_{i+1}$, then $C_i$ is the maximal element in $\mathcal{C}$ (the chain from which $\mathcal{C}^+$ is constructed). Hence we can take $C_i'$ to be $C_{i-1} \cup \{N + 1\}$ which belongs to the chain $\mathcal{C}^- \in SCP_{N+1}$, defined in (2.2), and $|\mathcal{C}^-| = |\mathcal{C}^+| - 2$ by construction.
- $C_{i-1}, C_i, C_{i+1}$ belong to a chain of the form $\mathcal{C}^- \in SCP_{N+1}$. Then there exists $\mathcal{C} \in SCP_N$ such that $\mathcal{C} = D_0 \lessdot \cdots \lessdot D_k$ and $C_j = D_j \cup \{N + 1\}$ for $0 \leqslant j \leqslant k - 1$. By inductive hypothesis, there exists a chain $\mathcal{D} \in SCP_N$ containing an element $D_i'$ with $D_{i-1} \lessdot D_i' \lessdot D_{i+1}$ and $|\mathcal{D}| = |\mathcal{C}| - 2$. Note that $D_{i+1}$ cannot be the maximal element, $D_k$, in $\mathcal{C}$, since $\mathcal{C}^- = D_0 \cup \{N + 1\} \lessdot \cdots \lessdot D_{k-1} \cup \{N + 1\}$. Therefore $D_i'$ is not the maximal element of $\mathcal{D}$. (If $D_i'$ were maximal then $|\mathcal{D}| \leqslant |\mathcal{C}| - 4$, since $D_i' \lessdot D_{i+1} \lessdot \cdots \lessdot D_k$ and, by Proposition 2.2.1, the length of a chain reduces by two when the size of the maximal element is reduced by one.) Thus $D_i' \cup \{N + 1\}$ belongs to $\mathcal{D}^- \in SCP_{N+1}$ and is the required element, since $|\mathcal{D}^-| = |\mathcal{D}| - 1 = |\mathcal{C}| - 3 = |\mathcal{C}^-| - 2$.

■

**Proof of Theorem 2.2.3** We wish to prove that for all $n \geqslant 2$,

$$2^\nu \leqslant |\mathcal{A}(2^{[n]})| \leqslant 3^\nu, \text{ where } \nu = \binom{n}{\lfloor n/2 \rfloor}.$$

In order to prove the left-hand side of the inequality, we note that $\mathcal{S} = \{S \subseteq [n] : |S| = \lfloor n/2 \rfloor\}$ belongs to $\mathcal{A}(2^{[n]})$, and every subset of $\mathcal{S}$ belongs to $\mathcal{A}(2^{[n]})$. There are $2^{|\mathcal{S}|} = 2^\nu$ subsets of $\mathcal{S}$.

The proof of the right-hand side of the inequality proceeds by counting the number of ways in which we can construct a filter $\mathcal{F}$ of $\langle 2^{[n]}, \subseteq \rangle$. The fact that there is a bijection between the set of filters and the set of antichains is proved in Corollary 6.1.1. We construct $\mathcal{F}$ by including zero or more elements from each chain in $SCP_n$ taking the chains in order of increasing length.

For chains of length at most two, we have at most three choices of elements to include in $\mathcal{F}$: neither element, the maximal element or the minimal element (and hence the maximal element as well).

Suppose that we now have to make a choice of elements from the chain $C_0 \lessdot C_1 \lessdot \cdots \lessdot C_k$ and that we have already chosen the elements from all chains of shorter length, thus fixing some part of $\mathcal{F}$. By Theorem 2.2.4 there exist $C_1', \ldots, C_{k-1}'$ such that $C_{i-1} \lessdot C_i' \lessdot C_{i+1}$ for $1 \leqslant i \leqslant k - 1$ and each $C'_i$ belongs to a shorter chain. By the construction of $\mathcal{F}$, we already know whether $C'_i \in \mathcal{F}$ for $1 \leqslant i \leqslant k-1$.

Define $l$ to be the largest index such that $C'_l \notin \mathcal{F}$ and $u$ to be the smallest index such that $C'_u \in \mathcal{F}$. (If $C'_i \in \mathcal{F}$ for $1 \leqslant i \leqslant k-1$ define $l = 0$ and if $C'_i \notin \mathcal{F}$ for $1 \leqslant i \leqslant k-1$ define $u = k$.) We have two possibilities: either

$$\underbrace{C'_1, \ldots, C'_l}_{\notin \mathcal{F}},\ \underbrace{C'_u, \ldots, C'_{k-1}}_{\in \mathcal{F}} \quad \text{and } u - l = 1\text{; or}$$

$$\underbrace{C'_1, \ldots, C'_{u-1}}_{\notin \mathcal{F}},\ \underbrace{C'_u}_{\in \mathcal{F}},\ \underbrace{C'_{u+1}, \ldots, C'_{l-1}}_{?},\ \underbrace{C'_l}_{\notin \mathcal{F}},\ \underbrace{C'_{l+1}, \ldots, C'_{k-1}}_{\in \mathcal{F}} \quad \text{and } u - l \leqslant -1.$$

Now $C_1, \ldots, C_{l-1} \notin \mathcal{F}$ since $C_1 \lessdot \cdots \lessdot C_{l-1} \lessdot C'_l$; and $C_{u+1}, \ldots, C_k \in \mathcal{F}$ since $C'_u \lessdot C_{u+1} \lessdot \cdots \lessdot C_k$. Hence $\mathcal{F}$ can only be extended by the inclusion of one or more of $C_l$ and $C_u$.

If $u - l = 1$ then

$$\underbrace{C_0 \lessdot \cdots \lessdot C_{l-1}}_{\notin \mathcal{F}} \lessdot C_l \lessdot C_u \lessdot \underbrace{C_{u+1} \lessdot \cdots \lessdot C_k}_{\in \mathcal{F}}$$

and hence we can make at most three choices to extend $\mathcal{F}$: we can choose neither $C_l$ nor $C_u$, choose $C_u$, or choose $C_l$ (which is equivalent to choosing both $C_l$ and $C_u$).

However, if $u - l \leqslant -1$ we cannot extend $\mathcal{F}$ without either duplication (since $C_{u+1} \leqslant C_l \in \mathcal{F}$) or violation of the conditions outlined in the preceding paragraph (since $C_u \leqslant C_{l-1} \notin \mathcal{F}$).

Hence for each of the $\nu$ chains we have at most three choices. The result follows. ■

Finally, we give two examples of the construction of a filter to illustrate the proof method in general, and how the value of $u - l$ affects the choice of elements from a chain. Tables 2.1 and 2.2 illustrate the construction used in the proof of Theorem 2.2.3. Column $\mathcal{C}$ contains the chains in $SCP_4$. Bold entries in this column indicate that an element has been selected from the chain for inclusion in the filter $\mathcal{F}$. Column $\mathcal{C}'$ contains the elements $C'_1, \ldots, C'_{k-1}$. Bold entries in this column indicate that an element has already been included in $\mathcal{F}$ by the construction to date. The next two columns indicate the elements $C_l$ and $C_u$, respectively, and the final column denotes the set of minimal elements in $\mathcal{F}$. We display only the minimal elements in $\mathcal{F}$ (denoted $\underline{\mathcal{F}}$) in order to conserve space.

For example, in row 4 of Table 2.1 we choose to include $\{3\}$, and hence $\{1,3\}$, in $\mathcal{F}$. As a result,

$$\mathcal{F} = \{\{3\}, \{1,3\}, \{2,3\}, \{3,4\}, \{1,2,3\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}\} \text{ and } \underline{\mathcal{F}} = \{3\}.$$

In the final row of Table 2.1 we have $u - l = -1$. Notice that $\{1,2,3\} \in \mathcal{F}$ since $\{3\} \in \mathcal{F}$, and that $\{1,2\}$ cannot be added to $\mathcal{F}$ since it is known that $\{1,2,4\} \notin \mathcal{F}$. In other words, as noted in the proof of Theorem 2.2.3, if $u - l \leqslant -1$ for some chain, then we cannot make any choices from that chain to extend $\mathcal{F}$.

**Proposition 2.2.2 (Hansel 1966)** *If $n$ is even, the result in Theorem 2.2.3 can be extended as*

| $\mathcal{C}$ | $\mathcal{C}'$ | $C_l$ | $C_u$ | $\underline{\mathcal{F}}$ |
|---|---|---|---|---|
| $\{2,4\}$ | – | – | – | $\emptyset$ |
| $\{\mathbf{3},\mathbf{4}\}$ | – | – | – | $\{\{3,4\}\}$ |
| $\{2\} \subset \{2,3\} \subset \{2,3,4\}$ | $\{2,4\}$ | $\{2,3\}$ | $\{2,3,4\}$ | $\{\{3,4\}\}$ |
| $\{\mathbf{3}\} \subset \{1,3\} \subset \{1,3,4\}$ | $\{\mathbf{3},\mathbf{4}\}$ | $\{3\}$ | $\{1,3\}$ | $\{\{3\}\}$ |
| $\{4\} \subset \{1,4\} \subset \{1,2,4\}$ | $\{2,4\}$ | $\{1,4\}$ | $\{1,2,4\}$ | $\{\{3\}\}$ |
| $\emptyset \subset \{1\} \subset \{1,2\} \subset \{1,2,3\} \subset \{1,2,3,4\}$ | $\{2\}, \{\mathbf{1},\mathbf{3}\}, \{1,2,4\}$ | $\{1,2,3\}$ | $\{1,2\}$ | $\{\{3\}\}$ |

Table 2.1: A filter construction in which $u - l \leqslant -1$ for a chain in $SCP_4$

| $\mathcal{C}$ | $\mathcal{C}'$ | $C_l$ | $C_u$ | $\underline{\mathcal{F}}$ |
|---|---|---|---|---|
| $\{2,4\}$ | – | – | – | $\emptyset$ |
| $\{\mathbf{3},\mathbf{4}\}$ | – | – | – | $\{\{3,4\}\}$ |
| $\{2\} \subset \{2,3\} \subset \{2,3,4\}$ | $\{2,4\}$ | $\{2,3\}$ | $\{2,3,4\}$ | $\{\{3,4\}\}$ |
| $\{\mathbf{3}\} \subset \{1,3\} \subset \{1,3,4\}$ | $\{\mathbf{3},\mathbf{4}\}$ | $\{3\}$ | $\{1,3\}$ | $\{\{3\}\}$ |
| $\{4\} \subset \{1,4\} \subset \{\mathbf{1},\mathbf{2},\mathbf{4}\}$ | $\{2,4\}$ | $\{1,4\}$ | $\{1,2,4\}$ | $\{\{3\},\{1,2,4\}\}$ |
| $\emptyset \subset \{1\} \subset \{\mathbf{1},\mathbf{2}\} \subset \{1,2,3\} \subset \{1,2,3,4\}$ | $\{2\}, \{\mathbf{1},\mathbf{3}\}, \{\mathbf{1},\mathbf{2},\mathbf{4}\}$ | $\{1\}$ | $\{1,2\}$ | $\{\{3\},\{1,2\}\}$ |

Table 2.2: A filter construction in which $u - l = 1$ for all chains in $SCP_4$

*follows:*

$$|\mathcal{A}(2^{[n]})| \leqslant 2^{(\nu-\mu)}3^{\mu}, \text{ \textit{where} } \mu = \binom{n}{\lfloor n/2 \rfloor - 1}.$$

**Proof** If $n$ is even, there are $\nu - \mu$ chains of length 1. Hence, from each of these chains, we can only make two choices instead of three. The result follows immediately. ■

## 2.3 Access control models

An access control model is a theoretical framework for reasoning about access control. An instance of an access control model has been referred to variously as a *system* (Bell and LaPadula 1973a; Harrison et al. 1976) or a *scheme* (Sandhu 1988; Sandhu 1992c). We will always use the term "system". A feature of any system is the idea of *state* (Bell and LaPadula 1973a) or *configuration* (Harrison et al. 1976) (of the system). We will use the term "state" throughout. The state of a system changes by the application of a *command* (Harrison et al. 1976; Sandhu 1992c) or *request* (Bell and LaPadula 1973a). We find both terms useful and will use them interchangeably. (The informal usage is that a request represents an attempt by a subject to access an object; a command attempts to change the authorizations of the system.) Systems have been treated as finite state machines in which the set of commands is used to define a transition relation (Bell and LaPadula 1973a) or have been analyzed by considering the effect of commands more directly (Harrison et al. 1976; Sandhu 1992c). We will adopt the latter approach in this thesis. In particular, our presentation of the Bell-LaPadula model is less formal than the original and does not use a transition relation.

## 2.3.1 The protection matrix model

The most widely known and commonly implemented access control model in commercial computer systems is the *protection matrix model.* It was introduced by Lampson (1971) and forms part of the Bell-LaPadula model (Bell and LaPadula 1973a) which we discuss in Section 2.3.2. The protection matrix model was further refined by Graham and Denning (1972). A simple form of the protection matrix model was introduced by Harrison et al. (1976) and has been the basis for subsequent research. We will use this form of the protection matrix model (or HRU model) throughout this thesis.

A protection matrix $M$ has columns indexed by objects and rows indexed by subjects. It is assumed that $S \subseteq O$. Each entry in $M$ is a subset of $A$, the set of access modes; $[s, o]$ denotes the entry in $M$ for subject $s$ and object $o$. Extremely simple examples of protection matrices are shown in Figure 2.5. We will also regard $M$ as a function $M : S \times O \to 2^A$, and write $M(s, o)$ to denote $[s, o]$.

Note that a column in the protection matrix represents an access control list for that object, and that a row in the protection matrix represents a capability list for that subject. In Figure 2.5a, for example, subject $s_1$ has the capability to `read` object $o_1$.

|  | $s_1$ | $s_2$ | $o_1$ | $o_2$ |
|---|---|---|---|---|
| $s_1$ |  |  | $\{\texttt{read}, \texttt{own}\}$ |  |
| $s_2$ |  |  |  | $\{\texttt{write}\}$ |

(a)

|  | $s_1$ | $s_2$ | $o_1$ | $o_2$ |
|---|---|---|---|---|
| $s_1$ |  |  | $\{\texttt{read}, \texttt{own}\}$ |  |
| $s_2$ |  |  | $\{\texttt{read}\}$ | $\{\texttt{write}\}$ |

(b)

**Figure 2.5:** Protection matrices

An *operation* changes $M$ by the insertion or deletion of either an object, a subject or an entry in the matrix. The six operations in the protection matrix model are `enter`, `create subject`, `create object`, `delete`, `destroy subject` and `destroy object`.[2] A *command* consists of an optional conditional statement and a body. A command is *conditional* if it contains a conditional statement, and is *unconditional* otherwise. The conditional statement tests one or more cells in the matrix for the *presence* of access rights. This is an example of fail-safe defaults. A conditional command may or may not change $M$ depending on the evaluation of the conditional statement. A command is *monotonic* if the body of the command consists only of enter and create operations. A command is *mono-operational* if the body of the command consists of a single operation. In this case the command is synonymous with an operation and we may refer to a `delete object` command, for example.

Command 2.1, for example, is a monotonic, conditional, mono-operational command. The command would succeed if the formal parameters, $s_o$ (owner), $s_f$ (friend) and $o$ were replaced

[2]The operations `enter` and `delete` add or remove rights from an entry in the matrix.

respectively by $s_1$, $s_2$ and $o_1$ in Figure 2.5a (and would fail otherwise): the resulting matrix is shown in Figure 2.5b.

**Command 2.1**

```
give-read($s_o, s_f, o$)
    if
        own $\in [s_o, o]$ and
        read $\in [s_o, o]$
    then
        enter read in $[s_f, o]$
```

A *protection system* $\mathsf{S}(A, \Gamma, M_0)$ is defined by a set of access rights $A$, a set of commands $\Gamma$ and an initial protection matrix $M_0 : S_0 \times O_0 \to 2^A$. When the interpretation of $A$, $\Gamma$ and $M_0$ are obvious from context we will simply write $\mathsf{S}$ to denote a protection system. A protection system is *monotonic* if each command is monotonic. In such a system the size of $M$ and the entries therein can only increase. A protection system is *mono-operational* if each command is mono-operational. A protection system is *mono-conditional* if each command is either unconditional or mono-conditional.

A *subject restricted protection system* is one in which no command includes the `create subject` operation (Lipton and Snyder 1978). A subject restricted protection system in which $|S_0| = k$ is called a *k-subject restricted protection system*.

Harrison et al. (1976) defined the *configuration* (state) of a protection system to be the triple $(S, O, M)$. One of the reasons why we treat $M : S \times O \to 2^A$ as a function is that $M$ completely defines the state of a protection system and is therefore more economical.

In summary, the HRU model imposes strict restrictions on the form of commands, but no restrictions on the commands themselves. This flexibility in the definition of commands means that the expressive power of the HRU model is unrivalled. Indeed, it has been shown that most access control models can be simulated using the HRU model for suitable choices of access rights and commands (Pittelli 1987; Sandhu 1992a; Sandhu 1992c; Snyder 1981). The price to be paid for this flexibility is that the HRU model has sufficient expressive power to simulate the behaviour of an arbitrary Turing machine (Harrison et al. 1976). This has significant implications for establishing the assurance of a system based on the HRU model. We discuss these issues in more detail in Chapter 5.

### 2.3.2 The Bell-LaPadula model

The act of accessing an object can be regarded as initiating a flow of information between subject and object. For example, read access can be seen as a flow of information from object to subject, while write access is a flow of information from subject to object. The Bell-LaPadula model implements an information flow policy designed to preserve the confidentiality of information.

We assume that an *information flow policy* is a lattice $\langle L, \to \rangle$, where $L$ is a set of labels. (The most general formalization of information flow policies (Denning 1976) does not assume that $L$ is a lattice. Sandhu (1992b) gives some examples of useful information flow policies that are not lattices.)

The ordering on the labels $\rightarrow$ determines the labels between which information can flow. $L$ is a lattice which means that information flow is reflexive, transitive and anti-symmetric. Furthermore, there exists a label that corresponds to public information (the bottom element of the lattice). Additionally, if $x \rightarrow z$ (information can flow from $x$ to $z$) and $y \rightarrow z$ (information can flow from $y$ to $z$), then $x \vee y$ exists and $x \vee y \rightarrow z$. Information flow policies (if correctly implemented) protect against deliberate attempts to compromise confidentiality (Trojan horses, for example) and accidental leakage of information (to insecure printing devices, for example).

The Bell-LaPadula model has been the subject of extensive research and rigorous analysis. It was developed in two seminal papers – Bell and LaPadula (1973a) and Bell and LaPadula (1973b). Every object and subject is associated with a label in the information flow policy. The anti-symmetry property requires that information must not leak from an object to a less secure subject, and information from a subject must not leak to a less secure object. The reference monitor in the Multics system (Bell and LaPadula 1976) was based on the Bell-LaPadula model as have been the reference monitors in many multi-level secure military systems.

Formally, the Bell-LaPadula model has the following characteristic features.

- A totally ordered set of *security classifications* or *security labels* $C$. That is, for all $c_1, c_2 \in C$ either $c_1 \leqslant c_2$ or $c_1 \geqslant c_2$. The most common set of security labels, used in military systems, is

$$\texttt{unclassified} < \texttt{classified} < \texttt{secret} < \texttt{top secret}. \tag{2.3}$$

- A set $K$ of (*needs-to-know*) *categories*. This set usually represents different projects or domains: $\{\texttt{navy}, \texttt{army}, \texttt{airforce}\}$, for example.

- A *security lattice* $L \subseteq C \times 2^K$, where $(c_1, K_1) \leqslant (c_2, K_2)$ if, and only if, $c_1 \leqslant c_2$ and $K_1 \subseteq K_2$. The security lattice represents the information flow policy.

- A set of *security clearance functions* $\Lambda$, where for all $\lambda \in \Lambda$, $\lambda : O \cup S \rightarrow L$. The essential idea of the Bell-LaPadula model is that every subject and object is assigned a security clearance by $\lambda$.

- A set of $|S| \times |O|$ protection matrices $\mathcal{M}$. (Unlike the protection matrix model, the set of objects is assumed to be fixed. New subjects are not created, existing ones are "activated".)

Figure 2.6 shows the security lattice, where $K = \{k_1, k_2\}$ and $C$ is given by (2.3). (In the figure we have abbreviated the security labels to a single initial letter because of space limitations.)

A Bell-LaPadula system $\mathsf{S}(A, C, K, \Gamma, M_0, \lambda_0, V_0)$ is determined by the set of access rights, the set of security labels, the set of needs-to-know classifications, the set of commands, the initial protection matrix, the initial security clearance function and the initial set of current authorizations (which is taken to be $\emptyset$). "Bell-LaPadula system" is not standard terminology; we adopt it because it is clear what we mean and it is consistent with the terminology in the thesis. Bell and LaPadula (1973b) developed their model by treating the set of commands as a transition relation in a finite state machine.

The *initial state* of a Bell-LaPadula system is the triple $t_0 = (V_0, M_0, \lambda_0)$. A Bell-LaPadula system evolves by the application of requests to the initial state. A state in general is a triple

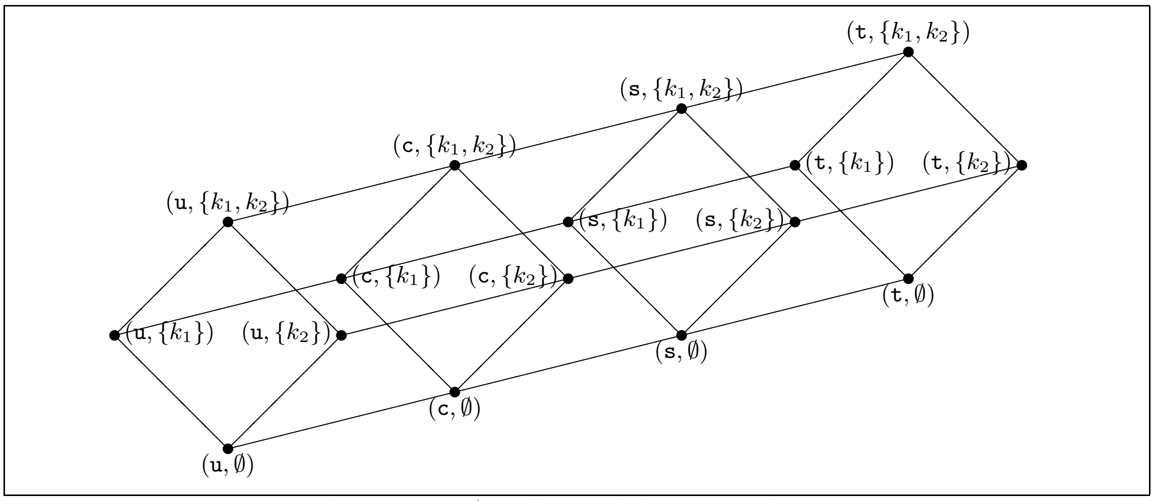


**Figure 2.6:** A Bell-LaPadula security lattice

$t = (V, M, \lambda)$. $V$ models the authorizations that have been granted and not yet revoked by the system ($V$ is taken from acti$\underline{\text{v}}$e). Note that $V$ models the active authorizations in $\mathsf{S}$, and is different from $M$ which models the potential authorizations in $\mathsf{S}$. $V$ is used when evaluating requests in order to ensure that the simple security property and *-property[3] are not violated, and is modelled as a set of triples of the form $(s, o, a)$, where $s \in S$, $o \in O$ and $a \in A$. We represent a state that results from the application of $i$ requests by $t_i = (V_i, M_i, \lambda_i)$. An *evolution* of $\mathsf{S}$ is a sequence of states $t_0 t_1 \ldots t_n$, which we denote $t^n$.

When particular access modes are introduced into the Bell-LaPadula model they are divided into two generic groups: read-type access modes, denoted $A_r$, and write-type access modes, denoted $A_w$. We will denote the set of *read triples* by $V_r$, and the set of *write triples* by $V_w$. Formally, $V_r = \{(s, o, a) \in V : a \in A_r\}$ and $V_w = \{(s, o, a) \in V : a \in A_w\}$, where $V_r \cup V_w = V$. It is not necessarily the case that $V_r \cap V_w = \emptyset$, since some access modes may be both read and write access modes. For example, in the system developed by Bell and LaPadula (1973b), the `write` access mode is interpreted as simultaneous read and write access, while the `read` and `append` access modes are read-only and write-only access modes, respectively.

This complicates the enforcement of the information flow policy. Read-type access modes present no problem on their own (see the simple security property below). However, indirect breaches of confidentiality can arise if a copy of an object can be written to a less secure object by a subject which has read access to the former and write access to the latter (see the *-property below).

### Simple security property

A state $t = (V, M, \lambda)$ satisfies the *simple security property* if, and only if,

$$\text{for all } v = (s, o, a) \in V_r,\ \lambda(s) \geqslant \lambda(o). \tag{2.4}$$

[3]The simple security property and *-property are defined below in (2.4) and (2.5), respectively.

In particular, if a request $\texttt{get}(s, o, a)$, where $a \in A_r$, is granted then $\lambda(s) \geqslant \lambda(o)$. (Otherwise, the resulting state will not satisfy the simple security property.)

The simple security property expresses the requirement that no user should be able to read an object that has a higher security clearance than the user.

### *-property

In order to more readily describe the *-property (read "star property"), we define

$$O(V, s) = \{o \in O : (s, o, a) \in V \text{ for some } a \in A\}.$$

That is, $O(V, s)$ is the set of objects in a state $V$ to which a particular subject $s$ currently has at least one access mode.

A state $t = (V, M, \lambda)$ satisfies the **-property* if, and only if,

$$\text{for all } s \in S,\ o_1 \in O(V_w, s),\ o_2 \in O(V_r, s),\ \lambda(o_1) \geqslant \lambda(o_2). \tag{2.5}$$

That is, a state satisfies the *-property if, and only if, for all subjects, the objects a subject can write to are at least as secure as the objects that subject can read.

The *-property expresses the requirement that no subject should be able to write (or "leak" information) to an object which is less secure than the subject. In particular, if the command $\texttt{get}(s, o, a)$, where $a \in A_w$, is granted then $\lambda(s) \leqslant \lambda(o)$ (otherwise there may be objects which $s$ can read which are more senior than $o$). The examples usually quoted for this rather counter-intuitive condition are

- to prevent Trojan horse software copying `top secret` material to an `unclassified` file, say;
- to prevent the writing of `top secret` material to an `unclassified` printer, for example.

Figure 2.7 shows a schematic representation of the evaluation of a `get` request for read access in the Bell-LaPadula model. If $\texttt{get}(s, o, \texttt{read})$ succeeds, then $(s, o, \texttt{read})$ is entered into $V$. (In practice, this evaluation process is more complicated (Bell and LaPadula 1973a), since the addition of a read triple may conflict with the *-property because of triples in $V_w$.)

### Secure systems

A state $t = (V, M, \lambda)$ is *secure* if $V$ satisfies the simple security property and the *-property. An evolution $t^n = t_0 t_1 \ldots t_n$, where $t_0$ is the initial state, is secure if $t_i$ is secure, $0 \leqslant i \leqslant n$. A system is secure if every evolution is secure. For the set of access rights $A = \{\texttt{read}, \texttt{append}, \texttt{write}, \texttt{execute}\}$ it is possible to write a set of security preserving commands $\Gamma$ (Bell and LaPadula 1973b).

**Theorem 2.3.1 (Basic Security Theorem (Bell and LaPadula 1973a))** *If $t_0$ is a secure state, then $\mathsf{S}(A, C, K, \Gamma, t_0)$ is a secure system.*

**Remark 2.3.1** *It is interesting to note that in the Bell-LaPadula paper the entries in the protection matrix are not required to satisfy the information flow policy. For example, it is possible that*

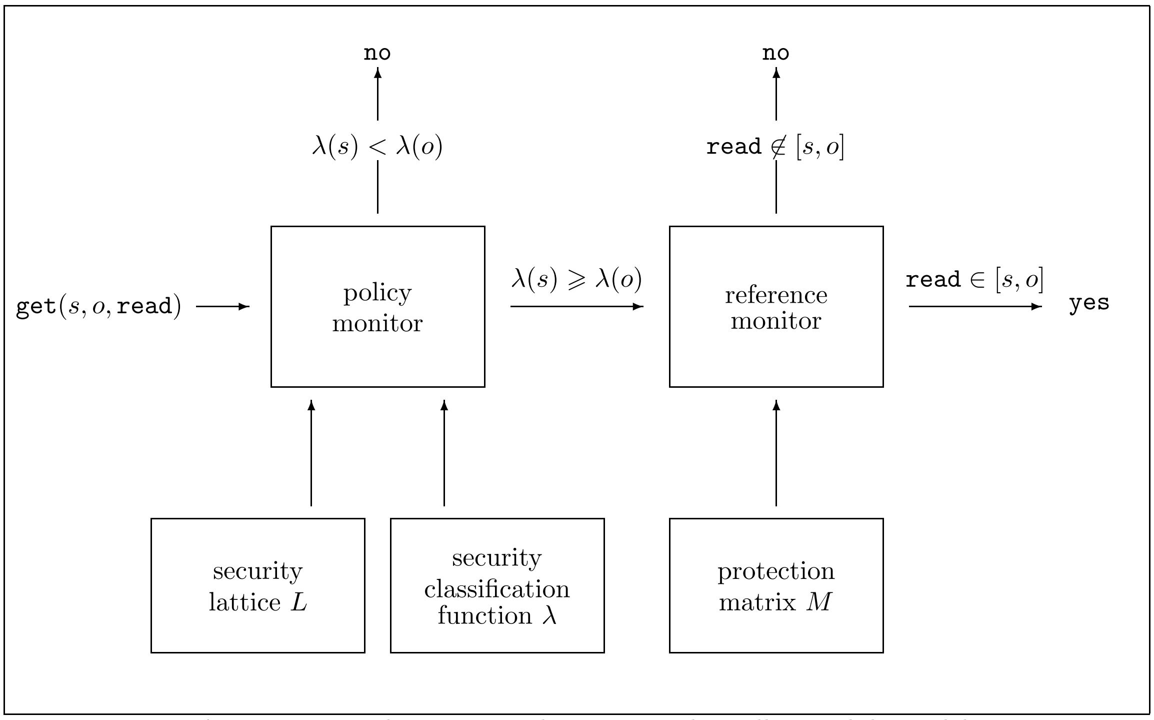

no
no
$\lambda(s) < \lambda(o)$
$\texttt{read} \notin [s, o]$
$\texttt{get}(s, o, \texttt{read})$
policy monitor
$\lambda(s) \geqslant \lambda(o)$
reference monitor
$\texttt{read} \in [s, o]$
yes
security lattice $L$
security classification function $\lambda$
protection matrix $M$

**Figure 2.7:** Evaluating a read request in the Bell-LaPadula model

*$s \in S$, $o \in O$, $a \in A_r$, where $\lambda(s) < \lambda(o)$ and $a \in [s, o]$. (This does not represent a breach of policy since the policy monitor will deny the request* $\texttt{get}(s, o, a)$*.)*

The Bell-LaPadula model implements an information flow policy designed to ensure the confidentiality of information. The Biba integrity model implements an information flow policy designed to ensure the integrity of information. We refer the interested reader to the literature for further details (Biba 1977).

The Bell-LaPadula model provides a rigorous definition of system security and, furthermore, demonstrates how to construct a secure system. In other words, the Bell-LaPadula model has strong security properties. Our ultimate goal, and the subject of Chapter 8, is to define a model that combines the strong security properties of the Bell-LaPadula model with the flexibility of the HRU model.

### Secure transitions

An alternative approach to secure systems was proposed by McLean (1990). The approach was motivated by exhibiting a system $\mathsf{Z}$ which starts in a secure state and, by definition, is a secure system, but which exhibits no desirable security properties. The reason for this is that the state transitions of $\mathsf{Z}$ are not secure in an intuitively reasonable sense (although they satisfy the requirements of the Bell-LaPadula model).

**Secure state transition function** Therefore, instead of defining what it means for a state to be secure, a *secure state transition function* is defined which ensures that any new state is reached in a secure manner. Specifically, it is not permitted for both $V$ and the security clearance function to be changed as a result of the state transition. A command $\gamma$ is *simple secure* if, and only if, $\gamma(V, M, \lambda) = (V', M', \lambda')$ implies

- if $(s, o, a) \in V' \setminus V$ and $a \in A_r$, then $\lambda = \lambda'$ and $\lambda(s) \geqslant \lambda(o)$;[4]
- if $\lambda(s) \neq \lambda'(s)$, for some $s \in S$, then
  - $\lambda(o) = \lambda'(o)$ for all $o \in O$,
  - $V = V'$,
  - for all $v \in V$ and for all $a \in A_r$, $(s, o, a) \in V$, $\lambda'(s) \geqslant \lambda'(o) = \lambda(o)$;
- if $\lambda(o) \neq \lambda'(o)$, for some $o \in O$, then
  - $\lambda(s) = \lambda'(s)$ for all $s \in S$,
  - $V = V'$,
  - for all $v \in V$ and for all $a \in A_r$, $(s, o, a) \in V$, $\lambda'(s) = \lambda(s) \geqslant \lambda'(o)$.

A corresponding definition is given for $\gamma$ to be **-secure.* A system is simple secure and *-secure if all commands in $\Gamma$ are simple secure and *-secure.

**Transition secure commands** McLean also suggested another approach in which we define a function $\psi : S \cup O \to 2^S$, where $\psi(o)$ is the set of subjects that can change the security clearance of $o$. A command $\gamma$ is *transition secure* if, and only if,

$$\gamma(s, (V, M, \lambda)) = (V', M', \lambda') \text{ implies for all } o \text{ such that } \lambda(o) \neq \lambda'(o),\ s \in \psi(o). \tag{2.6}$$

In other words, the subject $s$ must be authorized (by $\psi$) to change the security clearance of $o$. Two special cases arise: for all $o \in S \cup O$, $\psi(o) = \emptyset$; for all $o \in S \cup O$, $\phi(o) = S$. The first corresponds to a system in which $\lambda$ cannot be changed (such a system is said to have the property of *tranquillity*). The second corresponds to a system in which any subject can change $\lambda$ (and corresponds to the original Bell-LaPadula model). The notions of a transition being secure and *-property preserving are retained from the Bell-LaPadula model. A system is transition secure if every command in $\Gamma$ is transition secure.

## 2.4 Complexity theory

We first introduce the "big O" notation, which provides a convenient means of comparing the significant factor(s) affecting the values taken by functions. A function $\phi : \mathbb{N} \to \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers, is $\mathcal{O}(\psi(n))$ if there exist constants $c \in \mathbb{R}$ and $N \in \mathbb{N}$ such that $|\phi(n)| \leqslant c|\psi(n)|$ for all $n \geqslant N$.

A *decision problem* is one in which each instance of the problem has one of two answers. These answers are usually thought of as either `true` and `false`, `yes` and `no`, or 0 and 1. The safety problem, which we introduce in Chapter 5, is a decision problem.

[4]$\setminus$ denotes set difference; that is, $X \setminus Y = \{x \in X : x \notin Y\}$.

The *time complexity* (*function*) for an algorithm is a function $\tau : \mathbb{N} \rightarrow \mathbb{R}$ that expresses the longest time it would take for the algorithm to execute for an input of length $n$. (The space complexity function of an algorithm is defined in an analogous way.) A *polynomial time algorithm* is defined to be one whose time complexity function is $\mathcal{O}(p(n))$ for some polynomial function $p$, where $n$ denotes the length of the input to the algorithm. An algorithm whose time complexity function is not $\mathcal{O}(p(n))$ is said to be an *exponential time algorithm.* An *intractable* problem is one for which no polynomial time algorithm is known. A problem that can be solved in polynomial time by a non-deterministic algorithm is said to be **NP**-hard.

In a rigorous analysis of algorithmic complexity, it is necessary to fix an encoding scheme that will be used to determine the size of the input. In general, a more relaxed attitude is taken to the way in which the size of the input is measured. In broad terms, this merely results in different or less precise constant factors.

For example, the time complexity of an algorithm for sorting a list of integers into ascending order is simply expressed in terms of the length of the list rather than in terms of the length of the binary encoding of the list. (In this simple case, the length of the two different representations differs by a multiplicative factor $k$, where $k$ is the smallest number of bits required to represent every item in the list.) In short, we assume the time complexity function of an algorithm is independent of the encoding of the input. A more comprehensive treatment of complexity theory in general can be found in Greenlaw and Hoover (1998). An excellent guide to **NP**-hard problems can be found in Garey and Johnson (1979).

A problem for which no algorithm exists to solve all instances of the problem is called *undecidable.* The most well known undecidable problem is the *Halting problem* for Turing machines (Turing 1936) which is discussed in the following section.

Another important undecidable problem is the *Post correspondence problem* (Post 1946). Namely, given a set of pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$, where $s_i, t_i$ are strings over some alphabet, does there exist an integer $n$ and a set of integers $i_1, \ldots i_n$, $1 \leqslant i_j \leqslant k$, $1 \leqslant j \leqslant n$, such that the concatenation of the strings $s_{i_1}, \ldots, s_{i_n}$ equals the concatenation of the strings $t_{i_1}, \ldots, t_{i_n}$? Clearly there are instances of this problem which are decidable. For example consider the set of pairs $\{(01, 010), (01, 1), (1, 1)\}$. It is easiest to visualize the Post correspondence problem by treating each pair $(s_i, t_i)$ as a card with $s_i$ written above $t_i$ (Martin 1990). The correspondence problem then becomes: given an unlimited number of each of the $k$ cards, can they be arranged so that the top row matches the bottom row? Clearly for the set of pairs above we have the following solution:

$$\begin{array}{|c|c|c|} \hline 01 & 01 & 1 \\ 010 & 1 & 1 \\ \hline \end{array}\ .$$

However, the set of pairs $\{(01, 1), (01, 1), (1, 010)\}$ has no solution since the first symbol in each pair of strings is different. There is no algorithm which can decide all instances of the Post correspondence problem (Post 1946). The Post correspondence problem can be shown to be equivalent to the safety problem for a particular class of protection systems (Harrison and Ruzzo 1978).

## 2.5 Turing machines

A *deterministic Turing machine* (DTM) consists of a finite state control, a read-write head and a tape consisting of slots which can be indexed by $\mathbb{N}^+$. (Many formal descriptions of DTMs assume the tape is infinite in both directions; that is, the slots can be indexed by the set of integers $\mathbb{Z}$. It can shown that the computational power of either formulation is identical (Minsky 1967).) A *program* for a DTM consists of a set of tape *symbols* $\Sigma$, a finite set of *states* $Q$, and a *state transition function* $\delta$. There is a distinguished symbol $b$ called the *blank* symbol. $Q$ contains two distinguished states: the *start* state $q_0$ and the *halt* state $q_h$. The state transition function,

$$\delta : (Q \setminus \{q_h\}) \times \Sigma \to Q \times \Sigma \times \{-1, +1\},$$

takes as input a state and a symbol and returns a state, a symbol and a direction, where $-1$ denotes a move to the left and $+1$ a move to the right.

We illustrate the operation of a DTM through an example. If the machine is in state $q$ scanning the letter $s$ and $\delta(q, s) = (q', s', -1)$, then the machine enters state $q'$, overwrites $s$ with $s'$ and the read-write head moves one square to the left.

Given a DTM and a program, the *input* to the DTM is a string of symbols from $\Sigma$. One symbol from the input is entered (from left to right) in each square of the tape of the DTM starting at position 1. All other squares on the tape contain $b$. The program starts in state $q_0$ with the read-write head scanning position 1 of the tape. Thereafter, the operation of the machine is dependent on $\delta$ and the contents of the tape, and halts if the DTM enters state $q_h$. The *Halting problem* is undecidable (Turing 1936). That is, there is no algorithm to decide whether an arbitrary DTM enters the halt state $q_h$.

Finally we describe a convenient representation for a Turing machine at a particular instant. A *configuration* of a DTM is a string $x_0 \dots x_{i-1} q x_i \dots x_n$, where $x_0$ and $x_n$ are the leftmost and rightmost non-blank symbols on the tape, respectively, $q$ is the current state of the DTM, and the read-write head is scanning the square indexed by $i$, $0 \leqslant i \leqslant n$. The initial configuration of a DTM with input tape $x_0 \dots x_n$ is $q_0 x_0 \dots x_n$.

For further details about DTMs see Hopcroft and Ullman (1969) and Minsky (1967), for example. In Chapter 5 we will use the Halting problem to prove the undecidability of the safety problem for role-based access control.

# Chapter 3

# Role-based Access Control

Role-based access control has received considerable attention in recent years as an alternative to discretionary and mandatory access control models, particularly in open, distributed computing environments. The following quote provides a context and environment for the use of role-based access control.

> *"Role-based access control (RBAC) is a non-discretionary access control mechanism that allows and promotes the central administration of an organizational security policy. In many organizations in industry and civilian government, the end users do not 'own' the information for which they are allowed access. For these organizations, the corporation or agency is the actual 'owner' of system objects, and discretionary access control may not be appropriate. With RBAC, access control decisions are based on the roles individual users have as part of an organization."* Gligor (1995)

Gligor (1995) also identified the characteristics of an information system and organization that would make the system suitable for a role-based access control model. Briefly these are:

- a large number of users with largely generic job descriptions;
- a high turnover of staff and a high degree of mobility between jobs amongst the staff;
- a large number of data objects in applications which are broadly stable in terms of relationships between function and user and which are owned by the organization; and
- a requirement for organization-wide assessment of access control.

The main reason for the development of role-based access control was to address the perceived deficiencies of existing access control models and the complexity and cost of administering systems based on these models (Sandhu 1995). In particular, mandatory access control, based on the Bell-LaPadula model, was felt to be too restrictive, while discretionary access control, based on the protection matrix model, was not restrictive enough. In addition, in discretionary access control systems it is difficult to establish correctness of implementation because administration of access control is performed at the level of access control lists and/or capabilities, and does not provide easy mechanisms for managing constraints (O'Shea 1997).

Furthermore, discretionary access control mechanisms generally operate at the level of individual objects (Gligor 1995). This makes per-object review of access control straightforward but makes per-subject review difficult, if not impossible, to achieve in a large system.

Role-based access control models seek to improve and extend existing access control management functions in general, and focus on the management of permissions and roles, and of users and roles. It is claimed these functions are simpler in role-based access control than in traditional access control administration, provided the set of roles is relatively stable. Role-based access control also offers the prospect of per-subject access review. In particular, Gligor (1995) suggests that role-based access control has the potential to:

- reduce the effort and cost of security management;
- reduce the potential for error and confusion in security management;
- provide per-subject "before-the-act" audit functionality; and
- provide subject access profile update functionality.

In addition, it has been suggested that the administration of roles and permissions in role-based access control is performed at a level of abstraction that corresponds more naturally with the way in which an enterprise would deploy authorizations to employees (Ferraiolo et al. 1995).

A basic characterization of role-based access control was given in the call for papers of the First ACM Workshop on Role-based Access Control:

> *"... the essence of Role-Based Access Control (RBAC) is that rights and permissions are assigned to roles rather than to individual users. Users acquire these rights by virtue of being assigned membership in appropriate roles."*

The fundamental concepts of role-based access control are not new, using as they do many ideas familiar since the inception of multi-user computer and information systems. Since the late 1980s a number of papers have appeared using the idea of roles in access control (Ferraiolo and Kuhn 1992; Mohammed and Ditts 1994; Nyanchama and Osborn 1993; Nyanchama and Osborn 1994; Ting et al. 1992; von Solms and van der Merwe 1994).

Since the mid-1990s an attempt has been made to formalize these ideas in several different role-based access control models. The most well-known model, RBAC96, was introduced by Sandhu et al. (1996). It was followed by ARBAC97 (Sandhu et al. 1997), a model for role-based administration of roles. Further refinements to ARBAC97 were introduced by Sandhu and Munawer (1998b), and a complete description of ARBAC97 was published in Sandhu et al. (1999). The development and features of RBAC96 and ARBAC97 were influenced by a survey of user requirements regarding role-based access control conducted by the SETA corporation (Feinstein et al. 1995).

Since 1995 the formal development of role-based access control models has taken place in North America, notably at George Mason University, NIST and the SETA Corporation, and in Canada at the University of Western Ontario. While we will concentrate on role-based access control in general, and the RBAC96 family of models in particular, we mention two other projects on access control in distributed systems which incorporate roles and hence bear some similarity to role-based access control:

- Domain-based access control at Imperial College, London (Sloman 1994; Yialelis and Sloman 1996) uses object-oriented concepts for policy specification and deployment in distributed systems. The Ponder specification language (Damianou et al. 2000) includes `role` as a primitive data type. In this context, an instance of a role is a synonym for a set of Ponder policies.

- The OASIS model developed at the University of Cambridge provides secure interoperability of independent services in an open distributed architecture. Clients of a service are authenticated and thereby enter a service-specific role which can in turn be used as credentials to enter other roles provided by network services (Bacon et al. 2000; Hayton 1996; Hayton et al. 1998; Yao et al. 2001). OASIS bears a closer resemblance to role-based access control models than Ponder.

Three important role-based access control models have been developed: RBAC96/ARBAC97, the role graph model, and the NIST model. Aspects of each of these models have appeared in the course of the past ten years. However, no single description exists of the RBAC96/ARBAC97 model.[1] This chapter has three objectives:

- To present a comprehensive and uniform review of RBAC96 and ARBAC97, bringing together for the first time material from several seminal papers (Ahn and Sandhu 1999; Ahn and Sandhu 2000; Chen and Sandhu 1995; Sandhu et al. 1996; Sandhu et al. 1997; Sandhu and Munawer 1998b; Sandhu et al. 1999), and to summarize recent developments in role-based access control modelling in general. This review will motivate the improvements we suggest to role-based access control models in Chapter 4.

  We justify our decision to concentrate on RBAC96/ARBAC97 by showing that the role graph model (Nyanchama and Osborn 1999) and the NIST model (Gavrila and Barkley 1998) are equivalent to fragments of the RBAC96/ARBAC97 model.

- To introduce an intuitive, consistent, extensible notation for role-based access control.

  Each role-based access control model employs different terminology and notation. A significant feature of role-based access control models is the use of a role hierarchy, which can be modelled as a partially ordered set. Hence our notation is derived from the ideas introduced in Section 2.1. This approach has helped us to identify certain inconsistencies in the models we consider and to develop alternative suggestions.

- To consider the computational complexity of implementing RBAC96 and ARBAC97. Although Nyanchama and Osborn (1999) mention the complexity of implementing the role graph model, we are unaware of any similar work on the considerably more sophisticated and complex RBAC96/ARBAC97 model.

RBAC96 is a hierarchy of role-based access control models as shown in Figure 3.1. The interpretation of this hierarchy is that, for example, RBAC$_1$ includes all the features of RBAC$_0$. The simplest model, RBAC$_0$, defines the pre-requisite features of RBAC96 and role-based access

[1]We will refer to the RBAC96/ARBAC97 model, when we wish to emphasize that we are considering RBAC96 in conjunction with ARBAC97. We may also refer separately to the RBAC96 and ARBAC97 models.

control in general. RBAC$_1$ and RBAC$_2$ are not directly comparable, the former introducing the concept of role hierarchies, and the latter the concept of constraints on roles. RBAC$_3$ includes all the features of RBAC$_1$ and RBAC$_2$.

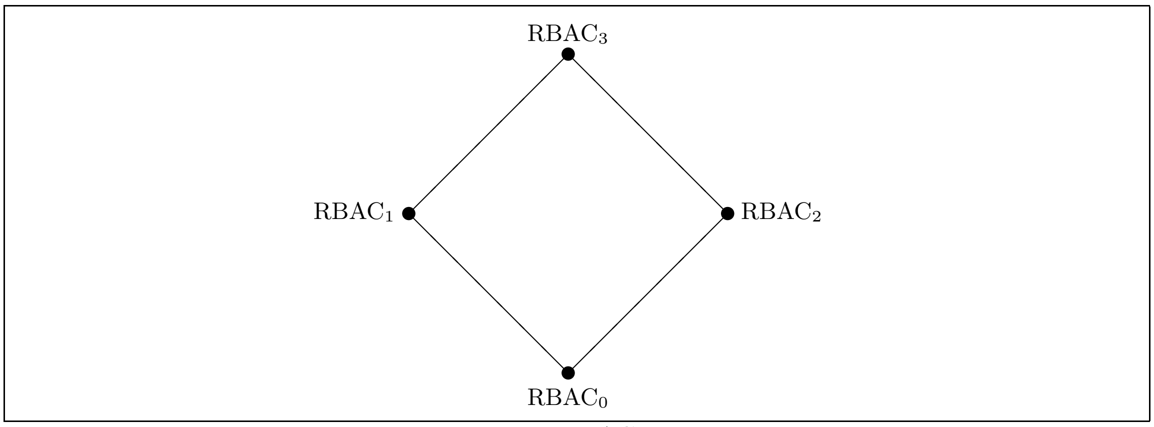


**Figure 3.1:** RBAC96 models

In the next section we present an overview of RBAC$_0$ and RBAC$_1$ concepts. In Section 3.2 we introduce a novel, formal notation for role-based access control, which is more economical and precise than is found in the literature. In Section 3.3 we consider RBAC$_2$ and the role constraint language RCL 2000. In Section 3.4 we describe the ARBAC97 model. In Section 3.5 the role graph model of Nyanchama and Osborn and the model developed at NIST are briefly discussed and compared to RBAC96. In Section 3.6 we briefly discuss the computational complexity of the more common procedures that would be required in an operational role-based access control system employing the RBAC96/ARBAC97 model.

The running example in this chapter and the remainder of the thesis was first used in Sandhu et al. (1997) to illustrate ARBAC97. The example has been augmented by several additional tables in order to improve the clarity of certain points. We have chosen to use this example rather than one of our own because we believe it provides a form of scientific control against which our contribution can be more accurately assessed.

## 3.1 RBAC$_1$

We denote the set of roles by $R$, the set of administrative roles by $AR$, and the set of permissions by $P$. Permissions are usually characterized by identifying an object and a set of access rights that are permitted for that object. (There is also the concept of a negative permission or *prohibition* which explicitly denies access to an object.) At this stage we will not further describe permissions, noting the following quote from the most recent attempt to create a standard for role-based access control.

*"The nature of a permission depends greatly on the implementation details of a system ... A general model for access control must therefore treat permissions as uninterpreted symbols ... "* (Ferraiolo et al. 2001)

**Definition 3.1.1** Permission-role assignment *is defined by the relation $PA \subseteq P \times R$, the semantics being that $(p, r) \in PA$ if permission $p$ is assigned to role $r$.*

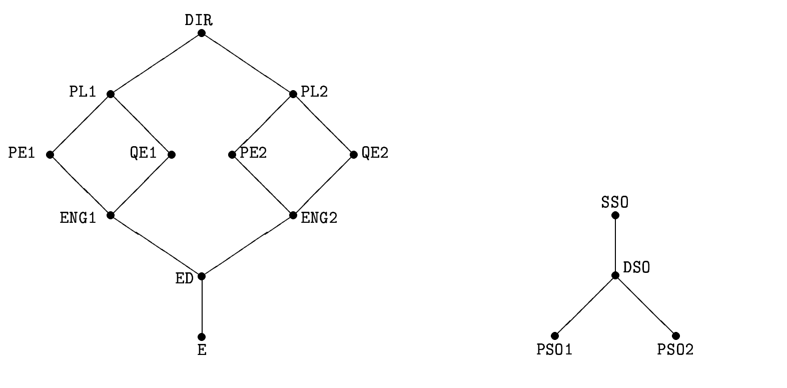
**Definition 3.1.2** User-role assignment *is defined by the relation $UA \subseteq U \times R$, the semantics being that $(u, r) \in UA$ if user $u$ is assigned to role $r$.*

**Groups vs. roles** We briefly identify why roles differ from groups in traditional access control paradigms. A group is a commonly used unit of access control consisting of a set of users. It is recognized that it is generally difficult to determine the permissions of a group, and hence a per-subject review of permissions is also difficult (O'Shea 1997). A role defines a group of users via the user-role assignment, and a group of permissions via the permission-role assignment. Hence a role is a unit of access control that establishes a connection between users and permissions, thereby making per-subject review far easier in principle.

**Definition 3.1.3** *A* role hierarchy*, $RH \subseteq R \times R$, is a reflexive, anti-symmetric, transitive binary relation.*

**Remark 3.1.1** *We note that we could regard $\langle R, \leqslant \rangle$ as a poset and interpret the role hierarchy as the covering relation of $R$. That is, given $r, r' \in R$, $(r, r') \in RH$ if, and only if, $r \lessdot r'$. This interpretation certainly has the attraction of economy when considering implementations of role-based access control. Indeed, the NIST model (Gavrila and Barkley 1998), which we discuss in Section 3.5.2, employs this approach, and is the one which we shall employ henceforth.*

A role hierarchy is represented by (the transitive reduction of) the graph of the relation $RH$. (In other words, a role hierarchy is represented by the Hasse diagram of the poset $\langle R, \leqslant \rangle$.) An example of a role graph (Sandhu et al. 1996) is shown in Figure 3.2a. This hierarchy will be used as the basis of all further examples in the thesis.



| | |
|---|---|
| DIR | Director |
| PL | Project Leader |
| PE | Production Engineer |
| QE | Quality Engineer |
| ENG | Engineer |
| ED | Engineering Department |
| E | Employee |
| SSO | Senior Security Officer |
| DSO | Departmental Security Officer |
| PSO | Project Security Officer |

(a) Role hierarchy (b) Administrative role hierarchy (c) Legend

**Figure 3.2:** RBAC96 hierarchies

In common with most approaches to role-based access control, we may say "role $r'$ is junior to role $r$" if $(r', r) \in RH$. However, we will usually employ more formal, concise and precise mathematical terminology and concepts. For example, we will say "the upper shadow of $r$" and write $\nabla r$ rather than "the set of all roles which are immediate seniors of role $r$".

The role hierarchy has two interpretations: the first with respect to permission inheritance, the second with respect to role activation.

**Permission Inheritance** It is assumed in RBAC$_1$ that a role $r$ is *implicitly* assigned all the permissions assigned to roles which are junior to $r$. The set $\{p \in P : (p, r) \in PA\}$ is the set of *explicit* permission assignments to $r$.

It is also assumed that if we have "negative" permissions (or *prohibitions*) then these are inherited downwards through the hierarchy. In other words, if $r' < r$ and $\overline{p} \in P(r)$, where $P(r)$ denotes the set of permissions explicitly assigned to $r$ and $\overline{p}$ denotes the explicit prohibition of permission $p$, then $\overline{p} \in P(r')$.

**Role Activation** It is assumed in RBAC$_1$ that if a user $u$ is explicitly assigned to a role $r$ (that is $(u, r) \in UA$), then $u$ can activate any role which is junior to $r$. All such roles are said to be *implicitly* assigned to $u$.

The existing literature does not discuss the possibility of *role exclusion* – the analogue of negative permissions, where, for example, $(u, \overline{r}) \in UA$ could be used to indicate that user $u$ cannot be assigned role $r$. We believe this would be a useful addition to the model. (In fact, it is possible to specify role exclusion using RCL 2000 or our model for conflict of interest policies. We discuss this further in Section 7.2.2.)

The RBAC$_1$ model implies that implicit user-role assignments are not included in $UA$ (Sandhu et al. 1996), although this suggestion is contradicted by the concept of weak revocation in the URA97 and PRA97 models (Sandhu et al. 1999). The implementation of the NIST model does not include implicit assignments in $UA$. This is clearly a sensible approach as it avoids duplication of information (since implicit assignments can be recovered from $UA$ and $RH$). However, we shall see that this has certain implications. For now we make the following observation.

**Remark 3.1.2** *If $(u, r_1), \ldots, (u, r_n) \in UA$ then we assume that $r_i \not< r_j$, $1 \leqslant i \neq j \leqslant n$ (otherwise $(u, r_i) \in UA$ is implied by $(u, r_j) \in UA$). More concisely, for all $u \in U$,*

$$R(u) \in \mathcal{A}(R),$$

*where $R(u)$ is the set of roles explicitly assigned to $u$, and $\mathcal{A}(R)$ is the set of antichains in the poset $R$.*

**Definition 3.1.4** *An* administrative role *is a role that includes administrative permissions to modify the sets of users, roles and permissions, and to modify the relations PA, UA and RH.*

The RBAC96 model requires that the set of roles $R$ and the set of administrative roles $AR$ be disjoint. Hence, there also exists an *administrative role hierarchy*, $ARH \subseteq AR \times AR$. An example is shown in Figure 3.2b.

The user-role assignment relation $UA$ is extended to include the set of administrative roles. That is, $UA \subseteq U \times (R \cup AR)$. Similarly, we define the permission-administrative-role relation $APA \subseteq AP \times AR$, where $AP$ is the set of administrative permissions and $AP$ is disjoint from $P$.

**Remark 3.1.3** *We will take the view that the set of (all) roles should be regarded as a partially ordered set. This set may be the disjoint union of two partial orders (that is, the set of normal roles and the set of administrative roles) if required.*

**Sessions** Users interact with a computer system (employing a role-based access control model) by establishing a *session* for which the user activates some subset of the roles to which the user is explicitly or implicitly assigned. We denote the set of sessions by $S$. The set of permissions available to a session is the "union of permissions from all active roles and all resulting implicit roles" (Sandhu et al. 1996).

A user may have more than one session running, where each session is a subset of roles to which the user is assigned. These roles may become active either through dynamic binding (that is, through the run-time environment) or through static binding (independent of the run-time environment).

A session in RBAC96 is analogous to a subject in traditional models of access control. In other words, access control decisions are made with reference to a particular session $s$ and hence with reference to the (permissions assigned to the) roles activated by $s$.

Sessions were introduced into RBAC96 partly to support the principle of *least privilege* (Saltzer and Schroeder 1975). That is, a subject (session) should only have available the permissions required to accomplish its task. Sessions enable a user to login and to invoke only those roles required in a given session. This is in contrast to many systems which enable all permissions of a user at login, irrespective of what the user is required to do in the course of that interaction with the computer system.

## 3.2 Notation for role-based access control

It is assumed in RBAC$_1$ that a role inherits permissions from the role-permission assignments of junior roles. Notice that $\{r' \in R : r' \leqslant r\}$ is $\downarrow r$ by definition. This motivates the notation in Table 3.1. This notation is not currently used in the role-based access control literature, but we believe it provides a more convenient and intuitive description of role-based access control properties and features. (Note that henceforth we will write "assigned" to mean "explicitly and implicitly assigned".)

| Notation | Mathematical Description | Semantics |
|---|---|---|
| $P(r)$ | $\{p \in P : (p, r) \in PA\}$ | the set of permissions explicitly assigned to $r$ |
| $\downarrow P(r)$ | $\{p \in P : (p, r') \in PA, r' \in \downarrow r\}$ | the set of permissions assigned to $r$ |
| $R(p)$ | $\{r \in R : (p, r) \in PA\}$ | the set of roles to which $p$ is explicitly assigned |
| $\uparrow R(p)$ | $\{r' \in R : (p, r) \in PA, r' \in \uparrow r\}$ | the set of roles to which $p$ is assigned |
| $R(u)$ | $\{r \in R : (u, r) \in UA\}$ | the set of roles explicitly assigned to $u$ |
| $\downarrow R(u)$ | $\{r' \in R : (u, r) \in UA, r' \in \downarrow r\}$ | the set of roles assigned to $u$ |
| $U(r)$ | $\{u \in U : (u, r) \in UA\}$ | the set of users explicitly assigned to $r$ |
| $\uparrow U(r)$ | $\{u \in U : (u, r') \in UA, r' \in \uparrow r\}$ | the set of users assigned to $r$ |

Table 3.1: A uniform, extensible notation for role-based access control

This notation is extensible in a natural way, so that $P(u)$, for example, would denote the permissions assigned explicitly to user $u$. Note that $\downarrow R(u)$ is an order ideal in $R$ and $R(u)$ is the set of maximal elements in this ideal. Similarly $\uparrow R(p)$ is an order filter.

We note that with the above definitions and notation, per-subject access review becomes a simple exercise. For a given user $u$, we compute the roles that $u$ can activate from $UA$ and hence the permissions $u$ has from $PA$. In particular, the permissions available to $u$ are given by

$$\downarrow P(u) = \bigcup_{r \in R(u)} \downarrow P(r) = \bigcup_{r \in \downarrow R(u)} P(r). \tag{3.1}$$

We can also analyze the use of sessions in RBAC96. We denote the user who established the session $s$ by $U(s)$, and the set of roles explicitly activated in $s$ by $R(s) \subseteq \downarrow R(u)$, where $u = U(s)$. We refer to $R(s)$ as an *active* set of roles. A user $u$ can activate any role in $\downarrow R(u)$. Then, by definition in RBAC96,

$$R(s) \subseteq \downarrow R(u) \quad \text{and} \quad P(s) = \{p \in P : (p, r) \in PA, r \in \downarrow R(s)\}.$$

We will assume, by analogy to $R(u)$, that $R(s) \in \mathcal{A}(R)$. Then we have

$$P(s) = \bigcup_{r \in R(s)} \downarrow P(r) = \bigcup_{r \in \downarrow R(s)} P(r),$$

which is analogous to (3.1).

We now extend our running example to include some examples of user-role and permission-role assignments. Tables 3.2, 3.3 and 3.4 do not appear in the literature. We have included them in order to make some of our points more explicit (particularly those related to Remark 3.1.2), and to motivate a discussion of the shortcomings of RBAC96 and ARBAC97.

Table 3.2 shows some typical user-role assignments. For each user $u$, $R(u) \in \mathcal{A}(R)$, where $R$ is the union of the role hierarchies depicted in Figure 3.2. (Clearly we have made some assumptions in this assignment of roles to users. For example, we have assumed that the Director would most likely be assigned the `SSO` role, and that the `PSO1` and `PSO2` roles would be assigned to Project Leaders. We note, however, that there is no constraint within the original example, that prevents `dave`, for example, being assigned the `SSO` role.)

In Table 3.3 we show some typical permission-role assignments, and in Table 3.4 we show possible choices for $R(s)$ and the corresponding permissions for a session $s$, when $U(s) =$ `bill`. (In the interests of brevity we have assumed that roles `ED` and `E` have no permissions assigned to them.)

| $u$ | $R(u)$ | $\downarrow R(u)$ |
|---|---|---|
| anne | {QE1, QE2} | {QE1, ENG1, ED, E, QE2, ENG2} |
| bill | {PL1, PSO1} | {PL1, PE1, QE1, ENG1, ED, E, PSO1} |
| claire | {DIR, SSO} | {DIR, PL1, ..., ED, E, PL2, ..., ENG2, SSO, DSO, PSO1, PSO2} |
| dave | {ENG1} | {ENG1, ED, E} |
| emma | {PE1, QE2} | {PE1, ENG1, ED, E, QE2, ENG2} |

Table 3.2: User-role assignments in RBAC96

| $r$ | $P(r)$ | $\downarrow P(r)$ |
|---|---|---|
| ENG1 | $\{p_1\}$ | $\{p_1\}$ |
| PE1 | $\{p_2\}$ | $\{p_1, p_2\}$ |
| QE1 | $\{p_3\}$ | $\{p_1, p_3\}$ |
| PL1 | $\{p_4\}$ | $\{p_1, p_2, p_3, p_4\}$ |

Table 3.3: Permission-role assignments in RBAC96

| $R(s)$ | $\downarrow R(s)$ | $P(s)$ |
|---|---|---|
| {ENG1} | {ENG1} | $\{p_1\}$ |
| {PE1} | {PE1, ENG1} | $\{p_1, p_2\}$ |
| {QE1} | {QE1, ENG1} | $\{p_1, p_3\}$ |
| {PE1, QE1} | {PE1, QE1, ENG1} | $\{p_1, p_2, p_3\}$ |
| {PL1} | {PL1, PE1, QE1, ENG1} | $\{p_1, p_2, p_3, p_4\}$ |

Table 3.4: Active roles and permissions in a session

## 3.3 RBAC$_2$: Constraints in role-based access control

RBAC$_2$ introduces the idea of constraints on roles enabling the articulation of *separation of duty policies* which specify two or more roles that one user cannot have active in the same session, for example; and *cardinality policies* which specify the maximum number of active instances of a role, for example. Informally, a policy in role-based access control is a set of *constraints* or *requirements*. We will use both terms interchangeably. Little work has been done on cardinality policies since their introduction, although they are supported in the NIST model. Therefore, this section concentrates on separation of duty policies.

### 3.3.1 Separation of duty policies

*Separation of duty* is the partitioning of a sensitive task into sub-tasks assigned to different users so that the co-operation of two or more users is required to complete the task. The purpose of separation of duty is to prevent a single user compromising the security requirements of an organization. A typical example in commercial environments is to require that one user prepares a cheque and a different user authorizes that cheque. In a military context, the launch of a missile, for example, may require two authorizations, one each from a different user (McLean 1990).

In role-based access control a separation of duty requirement is often modelled as a pair of roles. A distinction is made between *static* separation of duty, where the set of role assignments for each user must not contain both roles in the separation of duty requirement, and *dynamic* separation of duty, where the roles in each session must not contain both the roles in the separation of duty requirement (Barkley et al. 1997). Clearly static separation of duty is a stronger constraint on role assignment than dynamic separation of duty.

However, the structure of role-based access control admits several approaches to separation of duty. For example, *operational* separation of duty is the separation of duty at the level of permissions (Ferraiolo et al. 1995), where sensitive combinations of permissions are identified.

Such a combination of permissions cannot be assigned to any single user, even if the set of roles which have those permissions assigned to them do not conflict. Therefore, operational separation of duty is claimed to provide a higher level of assurance of separation of duty requirements than static separation of duty, and hence dynamic separation of duty (Ahn and Sandhu 1999). (Clearly we can differentiate between static and dynamic operational separation of duty. Indeed, several authors have gone to some trouble to identify different "flavours" of separation of duty (Gligor et al. 1998; Simon and Zurko 1997). However, although the level of granularity and implementation mechanism may be different, the underlying motivation and concept is the same in all cases.)

Separation of duty constraints may apply to a sequence of tasks $t_1, \ldots, t_k$, say, which are not performed in parallel. For example, if $t_1$ is the task "prepare cheque" and $t_2$ is "authorize cheque", most organizations would require that two different people perform these tasks. The consideration of separation of duty in workflow management systems (Bertino et al. 1999), although an important area for research, is not considered in RBAC$_2$ and is beyond the scope of this thesis.

### 3.3.2 RCL 2000

The language RSL99 was developed by Ahn and Sandhu (1999) to provide a framework for specifying role-based separation of duty and conflict of interest policies in role-based access control. RSL99 was subsequently revised, resulting in the role authorization constraint language RCL 2000 (Ahn and Sandhu 2000). In this section we describe the essential features of RCL 2000, discuss some of its drawbacks and suggest that first order predicate logic is equally suitable for expressing constraints, giving some examples to support this claim. An alternative approach based on Sperner families is presented in Chapter 7.

An important feature of RCL 2000 is the idea of a *conflicting set*.[2] In particular, we have the following components defined in RCL 2000.

- $\mathfrak{R} \subseteq 2^R$ – a set of conflicting role sets. An element, $\mathfrak{r} \subseteq R$, of this set defines a set of mutually exclusive roles which are used to articulate static separation of duty or dynamic separation of duty constraints.
- $\mathfrak{P} \subseteq 2^P$ – a set of conflicting permission sets. An element of this set, $\mathfrak{p} \subseteq P$, defines a set of mutually exclusive permissions which are used to articulate operational separation of duty constraints.
- $\mathfrak{U} \subseteq 2^U$ – a set of conflicting user sets. An element of this set, $\mathfrak{u} \subseteq U$, defines a set of conflicting users, which may be used to specify, for example, that two members of the same family cannot be assigned one each of two mutually exclusive roles.

$\mathfrak{R}$, $\mathfrak{P}$ and $\mathfrak{U}$ are used to define separation of duty requirements in the RBAC96 model. We will refer to an element in $\mathfrak{R}$ as a *separation of duty constraint*. We will usually use the set $\mathfrak{R}$ in our discussion of RCL 2000 and note that analogous observations can be made about $\mathfrak{P}$ and $\mathfrak{U}$.

RCL 2000 expressions are formed using the standard sets from the RBAC96 model, the sets $\mathfrak{R}, \mathfrak{P}$ and $\mathfrak{U}$, set operations and predicates (intersection, member of, etc) and two "non-deterministic functions" – `oneelement` (`OE`) and `allother` (`AO`). The purpose of these functions is

[2] A conflicting set is synonymous with a *separation of duty requirement* (Gavrila and Barkley 1998) or *conflict of interest constraint* (Nyanchama and Osborn 1999).

to eliminate logic quantifiers and thereby enable the intuitive articulation of constraints in RCL 2000. The semantics of an RCL 2000 expression are deduced by translating the expression into a restricted subset of first order predicate logic (RFOPL). A complete BNF specification for RCL 2000 and RFOPL can be found in Ahn and Sandhu (2000).

The algorithm `Reduction` translates an RCL 2000 expression into an RFOPL expression. The algorithm `Construction` translates an RFOPL expression into an RCL 2000 expression. Given an RCL 2000 expression $e$, $\mathtt{Construction}(\mathtt{Reduction}(e)) = e'$, where $e'$ is an RCL 2000 expression identical to $e$ up to naming of variables. An analogous result holds for $\mathtt{Reduction}(\mathtt{Construction}(e))$, where $e$ is an RFOPL expression.

The RFOPL expression[3]

$$\begin{aligned}&\forall\, cr \in CR, \forall\, r \in cr, \forall\, u \in U, \forall\, s \in sessions(u) : r \in roles(s) \Rightarrow \\ &(cr \setminus \{r\}) \cap roles(s) = \emptyset,\end{aligned} \tag{3.2}$$

expresses a dynamic separation of duty policy (Ahn and Sandhu 1999), and can be translated into the RCL 2000 expression

$$\begin{aligned}&\mathtt{OE}(\mathtt{OE}(CR)) \in roles(\mathtt{OE}(sessions(\mathtt{OE}(U)))) \Rightarrow \\ &(\mathtt{OE}(CR) \setminus \{\mathtt{OE}(\mathtt{OE}(CR))\}) \cap roles(\mathtt{OE}(sessions(\mathtt{OE}(U)))) = \emptyset.\end{aligned} \tag{3.3}$$

## 3.3.3 Comments on RCL 2000

RCL 2000 imposes no restrictions on the structure of $\mathfrak{R}$. However, we note that given $\mathfrak{R} \in 2^{2^R}$, the cardinality of $\mathfrak{R}$ is potentially very large. In Crampton et al. (1999) we observed that sets similar in structure to $\mathfrak{R}$ arise naturally in access control and can be reduced to an antichain in $\langle 2^R, \subseteq\rangle$. The results of our research in this area are presented in Chapter 7. We know by Hansel's result (Theorem 2.2.3) that $|\mathcal{A}(2^R)| \lll |2^{2^R}|$, where $\lll$ denotes significantly less than. For example, when $|R| = 4$, $|\mathcal{A}(2^R)| = 168$, whereas $|2^{2^R}| = 65536$.

The specification of RCL 2000 merely states that $\mathfrak{U}$ is a collection of sets of users, but also implies that such sets of conflicting users are defined with respect to particular conflicting roles. Therefore, it seems to make more sense to define $\mathfrak{U} \subseteq 2^U \times 2^R$, where $(\mathfrak{u}, \mathfrak{r}) \in \mathfrak{U}$ may be used to specify, for example, that two members of the same family cannot be assigned one each of two conflicting roles. This definition of $\mathfrak{U}$ differs from that in RCL 2000, as we have associated each set of conflicting users with a set of conflicting roles (which are not necessarily elements of $\mathfrak{R}$). Furthermore, using this formulation we can use constraints to specify role exclusion. For example, $(\mathfrak{u}, \{r\}) \in \mathfrak{U}$ specifies that the users in $\mathfrak{u}$ cannot be assigned to role $r$.

There seems little justification for the additional layer of abstraction introduced by RCL 2000, particularly as the meaning of an RCL 2000 expression is given by its reduction to RFOPL. We can only speculate that `oneelement` and `allother` were introduced to provide a blueprint for a non-deterministic algorithm for checking that constraints are satisfied.

[3]This expression uses the notation of RBAC96: $CR \equiv \mathfrak{R}$, $roles(s) \equiv R(s)$, and $sessions(u) \equiv S(u)$. The formulation of the same expression in our notation is shown in (3.4).

We believe that the first order expression (using our notation)

$$\forall\, s \in S, \forall\, \mathfrak{r} \in \mathfrak{R} : |\mathfrak{r} \cap R(s)| \leqslant 1 \tag{3.4}$$

is a perfectly readable and far more succinct formulation of (3.3).

We also note that an implicit assumption of RCL 2000, given the examples in Ahn and Sandhu (1999), is that no user can occupy more than one role in $\mathfrak{r} \in \mathfrak{R}$, as can be seen in expressions (3.2) and (3.3). We are, therefore, unsure as to why elements of $\mathfrak{R}$ in RCL 2000 can have an arbitrary number of elements rather than being mutually exclusive pairs, as in Barkley et al. (1997) and Gavrila and Barkley (1998), for example. In particular, it is far more efficient to check membership of conflicting role sets if they are doubleton sets rather than sets of arbitrary size (see Section 7.5).

If we grant that it may be desirable to have conflicting role sets of arbitrary size, then a user $u$ can be assigned to two or more roles in a conflicting role set $\mathfrak{r}$ provided $u$ is not assigned to *all* the roles in $\mathfrak{r}$. This formulation is shown in the expression

$$\forall\, s \in S, \forall\, \mathfrak{r} \in \mathfrak{R} : \mathfrak{r} \not\subseteq R(s).$$

Nevertheless, this point of view does not preclude the pairwise mutual exclusion implied in Ahn and Sandhu (1999), since we could replace $\mathfrak{R}$ by

$$\mathfrak{R}' = \bigcup_{\mathfrak{r} \in \mathfrak{R}} \bigcup_{\substack{r, r' \in \mathfrak{r} \\ r \neq r'}} \{r, r'\},$$

and use the first order expression

$$\forall\, s \in S, \forall\, \mathfrak{r} \in \mathfrak{R}' : |\mathfrak{r} \cap R(s)| \leqslant 1.$$

In other words, we remain unconvinced that RCL 2000 is any improvement on first order logic formulations of constraints, and that our formulation, particularly for $\mathfrak{U}$, is more expressive and no harder to understand. For example, we can easily formulate role exclusion constraints. (Cardinality constraints are not mentioned in either Ahn and Sandhu (1999) or Ahn and Sandhu (2000). Indeed, since the concept was introduced in Sandhu et al. (1996), there seems to have been little theoretical or practical interest in the concept except in Gavrila and Barkley (1998).)

## 3.4 ARBAC97

ARBAC97 is an administrative model that employs role-based access control concepts. The authors of ARBAC97 believe that the ability of role-based access control to reduce the administrative burden of access control in a large decentralized organization can usefully be employed in the administration of role-based access control itself. The model identifies three aspects of role-based administration that need consideration: administration of user-role assignment; administration of permission-role assignment; and administration of role-role assignment – that is, administration of the role hierarchy. These are addressed in three sub-models of ARBAC97: URA97 (Sandhu et al. 1997), PRA97 (Sandhu et al. 1997) and RRA97 (Sandhu et al. 1997; Sandhu and Munawer 1998b), respectively. Each of these models makes use of the administrative role hierarchy defined in RBAC96. With reference to the RBAC96 hierarchy of models (on page 44), we believe that URA97, PRA97 and RRA97 form ARBAC$_1$. Administration of constraints, the defining feature of RBAC$_2$, is not mentioned in ARBAC97. A complete description of ARBAC97 can be found in Sandhu et al. (1999).

### 3.4.1 URA97

URA97 is concerned with the management of the *UA* relation by administrative roles. Administrative roles are able to assign users to roles and revoke assignments of users to roles provided certain conditions are satisfied.

A *URA97 constraint* is defined recursively as follows:

- $r$ and $\overline{r}$ are constraints, where $r \in R$;
- $c_1 \wedge c_2$ and $c_1 \vee c_2$ are constraints, where $c_1$ and $c_2$ are constraints.

A constraint is evaluated with respect to a user $u$.[4] In particular,

- $r$ is satisfied if $r \in \downarrow R(u)$;
- $\overline{r}$ is satisfied if $r \notin \downarrow R(u)$;
- $c_1 \wedge c_2$ is satisfied if $c_1$ and $c_2$ are satisfied;
- $c_1 \vee c_2$ is satisfied if $c_1$ or $c_2$ is satisfied.

Intuitively, the constraint $r$ is satisfied by user $u$ if $r$ is assigned (either explicitly or implicitly) to $u$, and the constraint $\overline{r}$ is satisfied if $r$ is not assigned to $u$. Note that $r \vee \overline{r}$ is a valid URA97 constraint which is satisfied by all users.

URA97 defines two relations, `can-assign` $\subseteq AR \times \mathcal{C} \times \mathcal{R}$ and `can-revoke` $\subseteq AR \times \mathcal{R}$, where $\mathcal{C}$ is the set of constraints and $\mathcal{R}$ is the set of ranges in $R$. If $(a, c, R') \in$ `can-assign`, then $(u, r)$ can be added to $UA$ by the administrative role $a$ provided $r \in R'$ and $u$ satisfies constraint $c$. Similarly, if $(a, R') \in$ `can-revoke`, then $(u, r)$ can be removed from $UA$ by $a$ provided $r \in R'$.

Table 3.5 (Sandhu et al. 1997) shows examples of the `can-assign` and `can-revoke` relations with reference to Figure 3.2. For example, if we consider the `can-assign` relation, we see that `PSO1` can assign a user $u$ to roles `ENG1`, `PE1` and `QE1` (but not `PL1` because the upper end point is not included in the range) provided $u$ satisfies the constraint `ED`; that is, provided $u$ is assigned a role at least as senior as `ED`. Similarly, `DSO` can assign any user $u$ who satisfies the constraint `ED` to role `PL2` provided $u$ is not already assigned to any roles at least as senior as `PL1`.

Table 3.6 shows examples of users and whether each user satisfies the constraints of Table 3.5. (We use `T` to denote that a constraint is satisfied, and `F` otherwise.)

Table 3.7 shows some examples of user-role assignment in URA97: $r$ denotes a role which can be assigned to a user $u$; $A$ denotes the set of administrative roles that can assign $r$ to $u$ (given the

[4]In order to indicate the utility and economy of our notation we note that the original formulation in Sandhu et al. (1997) is as follows: $r$ is satisfied if there exists $r' \geqslant r$ such that $(u, r') \in UA$; and $\overline{r}$ is true if for all $r' \geqslant r, (u, r') \notin UA$.

`can-assign` relation in Table 3.5); and $R'(u)$ denotes the resulting explicit role assignments to $u$. Note that `bill` cannot be assigned to `PL2` because only row 3 of the `can-assign` relation admits assignment of users to `PL2`, and from Table 3.6 we see that `bill` does not satisfy the constraint $\texttt{ED} \wedge \overline{\texttt{PL1}}$. However, he can be assigned to roles less senior than `PL2`. Note also that the new user, `fred`, cannot be assigned to any roles because he does not satisfy any constraints. Furthermore, `dave` can be assigned to roles considerably more senior than the one he is currently assigned.

| can-assign | | |
|---|---|---|
| Administrative Role | Constraint | Role Range |
| PSO1 | ED | [ENG1, PL1) |
| PSO2 | ED | [ENG2, PL2) |
| DSO | $\texttt{ED} \wedge \overline{\texttt{PL1}}$ | [PL2, PL2] |
| DSO | $\texttt{ED} \wedge \overline{\texttt{PL2}}$ | [PL1, PL1] |

| can-revoke | |
|---|---|
| Administrative Role | Role Range |
| PSO1 | [ENG1, PL1) |
| PSO2 | [ENG2, PL2) |
| DSO | [ED, DIR] |

Table 3.5: URA97 relations

| $u$ | $R(u)$ | Constraint | | | | |
|---|---|---|---|---|---|---|
| | | ED | $\overline{\texttt{PL1}}$ | $\overline{\texttt{PL2}}$ | $\texttt{ED} \wedge \overline{\texttt{PL1}}$ | $\texttt{ED} \wedge \overline{\texttt{PL2}}$ |
| anne | {QE1, QE2} | T | T | T | T | T |
| bill | {PL1} | T | F | T | F | T |
| dave | {ENG1} | T | T | T | T | T |
| fred | $\emptyset$ | F | T | T | F | F |

Table 3.6: Users and constraints in URA97

| $u$ | $R(u)$ | $r$ | $A$ | $R'(u)$ |
|---|---|---|---|---|
| anne | {QE1, QE2} | PE1 | {PSO1, DSO} | {QE1, QE2, PE1} |
| anne | {QE1, QE2} | PL1 | {DSO} | {PL1, QE2} |
| bill | {PL1} | PE2 | {PSO2, DSO} | {PL1, PE2} |
| dave | {ENG1} | PL1 | {DSO} | {PL1} |

Table 3.7: User-role assignments in URA97

### User-role revocation

Revocation of a user-role assignment in URA97 is defined to be *weak* by default, meaning that if user-role assignment $(u, r) \in UA$ is revoked but $(u, r') \in UA$, where $r < r'$, then $u$ can still exercise the permissions of $r$ through $\downarrow P(r')$. *Strong* revocation is accomplished by performing successive weak revocations provided such revocations are allowed by the `can-revoke` relation (Sandhu et al. 1999). Weak revocation, therefore, rests on the assumption that the set of roles assigned to a user is not an antichain. That is, RBAC96 allows a user to be both explicitly and implicitly assigned to a role. In short, insisting that $R(u) \in \mathcal{A}(R)$ is incompatible with weak revocation in the URA97 model.

We believe the motivation for weak revocation is not sufficiently convincing to warrant the introduction of such an overhead into the model. It is hard to envisage situations in which weak revocation would be useful. Indeed, a characteristic feature of the RBAC96 model is the use of a role hierarchy to define implicit user- and permission-role assignments.

We believe a more realistic situation to model would be the transfer of a user $u$ from a role $r$ to a more senior one $r'$, in the case of promotion, for example. A transfer would be modelled as the revocation of $(u, r)$ and the assignment of $(u, r')$.

#### Updates to the *UA* relation

Suppose that $u$ is assigned to $r$ by administrative role $a$. Suppose further that there exists $r' \in R$ such that $(u, r') \in UA$ and $r' < r$. Then for $R(u)$ to be an antichain there must be a revoke operation as well as the assign operation. Specifically, $(u, r')$ should be removed from $UA$. It is not clear from the definition of URA97 whether it is required that $(a, R'') \in$ `can-revoke` as well as $(a, c, R') \in$ `can-assign`, where $r' \in R''$, $r \in R'$ and $u$ satisfies $c$.

### 3.4.2 PRA97

PRA97 is concerned with the management of the $PA$ relation. Administrative roles are able to assign permissions to roles and revoke assignments of permissions to roles subject to certain conditions being satisfied. A *PRA97 constraint* has the same structure as a URA97 constraint but is evaluated with respect to a permission, $p$. Specifically we have

- $r$ is satisfied if $r \in {\uparrow}R(p)$ (note that this is an inversion of the condition for the satisfaction of $r$ in $URA97$);
- $\overline{r}$ is satisfied if $r \notin {\uparrow}R(p)$;
- $c_1 \wedge c_2$, where $c_1$ and $c_2$ are constraints, is satisfied if $c_1$ and $c_2$ are satisfied;
- $c_1 \vee c_2$, where $c_1$ and $c_2$ are constraints, is satisfied if $c_1$ or $c_2$ is satisfied.

Intuitively, the constraint $r$ is satisfied by a permission $p$ if $p$ is assigned (either explicitly or implicitly) to $r$, and the constraint $\overline{r}$ is satisfied if $p$ is not assigned to $r$.

PRA97 defines two relations `can-assignp` $\subseteq AR \times \mathcal{C} \times \mathcal{R}$ and `can-revokep` $\subseteq AR \times \mathcal{R}$ which are analogous to the URA97 relations `can-assign` and `can-revoke`, respectively. Specifically, if $(a, c, R') \in$ `can-assignp`, then $(p, r)$ can be added to $PA$ by the administrative role $a$ provided $r \in R'$ and $p$ satisfies constraint $c$; and if $(a, R') \in$ `can-revokep`, then the administrative role $a$ can remove $(p, r)$ from $PA$, provided $r \in R'$.

The relations in Table 3.8 are taken from Sandhu et al. (1997). `PSO1`, for example, can assign permissions that are assigned to `PL1` (and not to `QE1`) to `PE1`. Similarly, `PSO1` can assign permissions already assigned to `PL1` (and not to `PE1`) to `QE1`. In particular, `PSO1` cannot assign the same permission to both `PE1` and `QE1`.

#### Permission-role revocation

Revocation of a permission-role assignment in PRA97 is defined to be *weak* by default, meaning that if permission-role assignment $(p, r) \in PA$ is revoked but $(p, r') \in PA$, where $r' < r$, then $r$ can still exercise the permission $p$ through ${\uparrow}P(r')$. Strong revocation is accomplished by performing successive weak revocations provided such revocations are allowed by the `can-revokep`

| can-assignp | | |
|---|---|---|
| Administrative Role | Constraint | Role Range |
| DSO | DIR | [PL1, PL1] |
| DSO | DIR | [PL2, PL2] |
| PSO1 | $\mathtt{PL1} \wedge \overline{\mathtt{QE1}}$ | [PE1, PE1] |
| PSO1 | $\mathtt{PL1} \wedge \overline{\mathtt{PE1}}$ | [QE1, QE1] |
| PSO2 | $\mathtt{PL2} \wedge \overline{\mathtt{QE2}}$ | [PE2, PE2] |
| PSO2 | $\mathtt{PL2} \wedge \overline{\mathtt{PE2}}$ | [QE2, QE2] |

| can-revokep | |
|---|---|
| Administrative Role | Role Range |
| DSO | [ED, DIR] |
| PSO1 | [QE1, QE1] |
| PSO1 | [PE1, PE1] |
| PSO2 | [PE2, PE2] |
| PSO2 | [QE2, QE2] |

Table 3.8: PRA97 relations

relation (Sandhu et al. 1997). In short, insisting that $R(p) \in \mathcal{A}(R)$ is incompatible with the PRA97 revoke model.

**Updates to the *PA* relation**

Note that if $(p, \mathtt{PL1}) \in PA$ and $p$ is assigned to PE1, then by virtue of inheritance and the fact that only explicit assignments are held in the $PA$ relation, assigning $(p, \mathtt{PE1})$ to $PA$ should also revoke $(p, \mathtt{PL1})$.

In general, suppose that $p$ is assigned to $r$ by role $a$. Suppose further that there exists $r' \in R$ such that $(p, r') \in PA$ and $r < r'$. Then for $P(r)$ to be an antichain $(p, r') \in PA$ must be revoked. It is not clear from the definition of PRA97 whether it is required that $(a, R'') \in$ can-revokep as well as $(a, c, R') \in$ can-assignp, where $r' \in R''$, $r \in R'$ and $p$ satisfies $c$.

## 3.4.3 RRA97

RRA97 was introduced in the same paper as URA97 and PRA97 and then significantly extended. RRA97 considers the administration of the role hierarchy. We first present the example from Sandhu et al. (1997) which provides the motivation for many of the refinements in the mature RRA97 model (Sandhu and Munawer 1998b).

**Example 3.4.1** Suppose DSO inserts roles X and Y into the hierarchy, creating the edges $(\mathtt{X}, \mathtt{DIR})$, $(\mathtt{QE1}, \mathtt{X})$ and $(\mathtt{Y}, \mathtt{PE1})$. Suppose in addition that PSO1 subsequently inserts the edge $(\mathtt{PE1}, \mathtt{QE1})$. The resulting hierarchy is shown in Figure 3.3a, in which new edges are indicated by dotted lines. The transitive reduction of this hierarchy is shown in Figure 3.3b.

Sandhu et al. (1997) observe that by transitivity we now have $\mathtt{X} > \mathtt{Y}$, when, in fact, X and Y were incomparable roles on their creation, and that this is an "undesirable side effect" of unrestricted changes to the role hierarchy.[5] They go on to suggest: either DSO should not be able to create roles X and Y, as they conflict with the existing authority range of PSO1; or that PSO1 should not be able to make PE1 less than QE1; or that such a possibility should be regarded as acceptable. RRA97 is developed with the first of these options in mind, thereby seeking to maximize the potential for decentralization of administration and autonomy of administrative roles (Sandhu et al. 1997).

[5]An "undesirable side effect" is never formally defined in RRA97. We also note that since PE1 and QE1 are also incomparable to start with and that subsequently $\mathtt{PE1} < \mathtt{QE1}$, we are unsure why the fact that $\mathtt{Y} < \mathtt{X}$ should be considered undesirable.

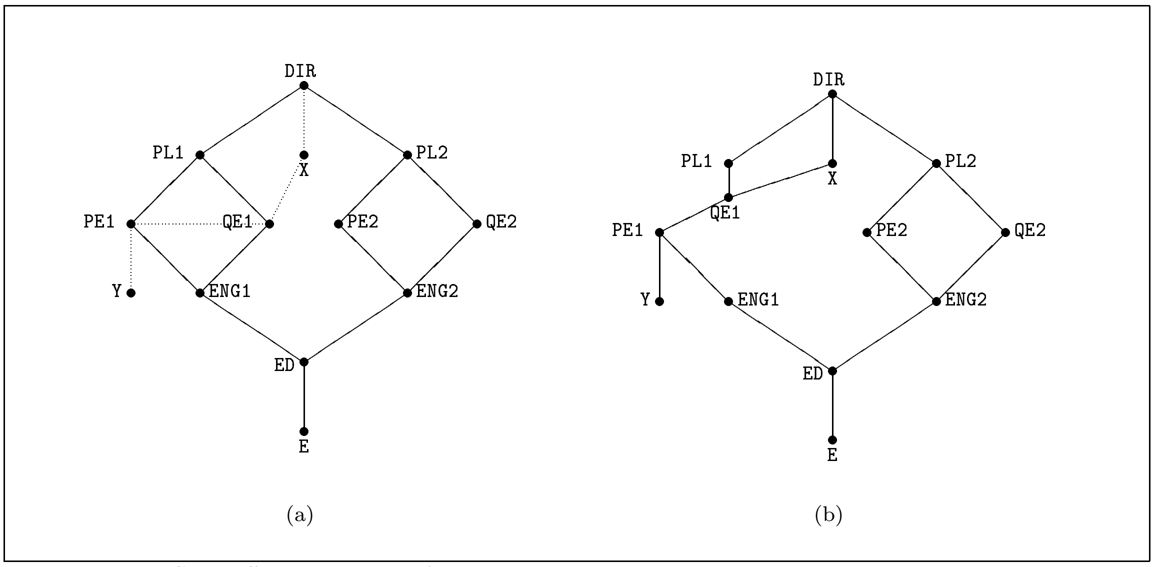


**Figure 3.3:** Side effects resulting from changes to the role hierarchy in Figure 3.2: The roles `X` and `Y` and the edges (`Y`, `PE1`), (`PE1`, `QE1`), (`QE1`, `X`) and (`X`, `DIR`) have been added to the hierarchy

RRA97 addresses the following considerations, which we will refer to as *role hierarchy operations* or simply *hierarchy operations*: role insertion, role deletion, edge insertion and edge deletion. The model also addresses the following problems which could arise as a result of the four hierarchy operations.

- No cycles should be introduced into the hierarchy. (That is, $R$ should remain a partial order.)
- Successive changes to the role hierarchy "should not lead to undesirable side effects". An example of this is `Y` becoming less than `X` as a result of the insertion of an edge in the role hierarchy.
- What is the effect of role hierarchy operations on the other relations in ARBAC97? (This includes, in particular, role deletion and its effect on ranges in tuples in the `can-` relations in URA97 and PRA97.)
- How should permissions and users that are assigned to a deleted role be dealt with?

The central idea in RRA97 is the relation `can-modify` $\subseteq AR \times \mathcal{R}$. If $(a, R') \in$ `can-modify`, then the administrative role $a$ can make changes to the hierarchy within the range $R'$. $R'$ is referred to as an *authority range*. An example of the `can-modify` relation (Sandhu et al. 1997) is shown in Table 3.9.

| `can-modify` | |
|---|---|
| Administrative Role | Authority Range |
| `PSO1` | (`ENG1`, `PL1`) |
| `DSO` | (`ED`, `DIR`) |

Table 3.9: The `can-modify` relation

Authority ranges must satisfy three properties. Firstly, an authority range must be an open range. Secondly, given two authority ranges $R', R'' \subseteq R$, either $R' \cap R'' = \emptyset$ or $R' \subseteq R''$ or $R'' \subseteq R'$. For example, in Table 3.9 $(\texttt{ENG1}, \texttt{PL1}) \subset (\texttt{ED}, \texttt{DIR})$. Thirdly, an authority range must be *encapsulated.* The intuition behind the notion of an encapsulated range is that "all the roles in such a range have an identical relation to roles outside the range". Furthermore, an encapsulated range "is the correct unit for autonomous management of role-role relationships with the range" (Sandhu and Munawer 1998b). This statement is justified by the observation that decentralization of authority and autonomy requires that all inward and outward edges (in a role hierarchy) from an authority range should be directed to and from the end points of the authority range. In short, an encapsulated range can be considered to be an autonomous sub-hierarchy within which hierarchy operations may be performed.

**Definition 3.4.1 (Sandhu and Munawer 1998b)** *A range $(x, y)$ is said to be* encapsulated *if for all $w \in (x, y)$, and for all $z \notin (x, y)$,*

$$z > w \text{ if, and only if, } z > y, \text{ and} \tag{3.5}$$

$$z < w \text{ if, and only if, } z < x. \tag{3.6}$$

**Remark 3.4.1** *We note that with this definition no range can be encapsulated since $x, y \notin (x, y)$, $y > w$ for all $w \in (x, y)$ and $x < w$ for all $w \in (x, y)$. Hence conditions* (3.5) *and* (3.6) *should be replaced by*

$$z > w \text{ if, and only if, } z \geqslant y \text{ and} \tag{3.7}$$

$$z < w \text{ if, and only if, } z \leqslant x, \tag{3.8}$$

*respectively.*

However, we prefer the following succinct characterization of an encapsulated range using our notation. In fact, Proposition 3.4.1 provides the motivation for our model of administration in Chapter 4.

**Proposition 3.4.1** *A range $(x, y)$ is encapsulated if, and only if,*

$$\uparrow(x, y) \setminus \uparrow y = (x, y) \text{ and} \tag{3.9}$$

$$\downarrow(x, y) \setminus \downarrow x = (x, y). \tag{3.10}$$

**Proof**

⇒ Suppose for all $z \notin (x, y)$ and for all $w \in (x, y)$ we have $z > w$ if, and only if, $z \geqslant y$. We now prove that $\uparrow(x, y) \setminus \uparrow y \subseteq (x, y)$. Let $a \in \uparrow(x, y) \setminus \uparrow y$. Then $y \not\leqslant a$ and there exists $b \in (x, y)$ such that $b \leqslant a$. Since $(x, y)$ is encapsulated, $a \in (x, y)$ (otherwise we have $a \notin (x, y)$ such that $a \geqslant b$ for some $b \in (x, y)$ and $a \not\geqslant y$). That is $\uparrow(x, y) \setminus \uparrow y \subseteq (x, y)$. Clearly, $(x, y) \subseteq \uparrow(x, y) \setminus \uparrow y$ and hence we have $\uparrow(x, y) \setminus \uparrow y = (x, y)$.

The corresponding proof for $\downarrow(x, y) \setminus \downarrow y$ is similar; we omit the details.

$\Leftarrow$ Suppose $\uparrow(x, y) \setminus \uparrow y = (x, y)$. Let $w \in (x, y)$ and $z \notin (x, y)$ with $z > w$. Hence $z \in \uparrow(x, y)$. Since $z \notin (x, y)$, $z \in \uparrow y$ and hence $z \geqslant y$.

The corresponding proof for $\downarrow(x, y) \setminus \downarrow y$ is similar; we omit the details.

■

Table 3.10 shows the possible ranges for the left-hand side of the role hierarchy in Figure 3.2a. The table entry in row $x$ and column $y$ is the set of elements in the (open) range $(x, y)$. The encapsulated ranges are starred.

| End points | ENG1 | PE1 | QE1 | PL1 | DIR |
|---|---|---|---|---|---|
| ED | | {ENG1} | {ENG1} | {ENG1, PE1, QE1}$^*$ | {ENG1, PE1, QE1, PL1}$^*$ |
| ENG1 | | | | {PE1, QE1}$^*$ | {PE1, QE1, PL1}$^*$ |
| PE1 | | | | | {PL1} |
| QE1 | | | | | {PL1} |
| PL1 | | | | | |

Table 3.10: Open ranges and encapsulated ranges

We now define two further types of range which are central to edge and role insertion in RRA97. The *immediate authority range* of a role $r$, denoted $(r)$, is the smallest authority range that contains $r$. Formally, $(r)$ is the authority range $(x, y)$ such that $r \in (x, y)$ and for all authority ranges $(x', y') \subset (x, y)$, $r \notin (x', y')$. Note that the set of immediate authority ranges is a subset of the set of authority ranges which is a subset of the set of encapsulated ranges. Suppose now that (ENG1, PL1) and (ED, DIR) are authority ranges as shown in Table 3.9. Then Table 3.11 shows the corresponding immediate authority ranges for each role. (A question mark indicates the immediate authority range is not defined.)

| $r$ | $(r)$ |
|---|---|
| ED | ? |
| ENG1 | (ED, DIR) |
| PE1 | (ENG1, PL1) |
| QE1 | (ENG1, PL1) |
| PL1 | (ED, DIR) |
| DIR | ? |

Table 3.11: Immediate authority ranges

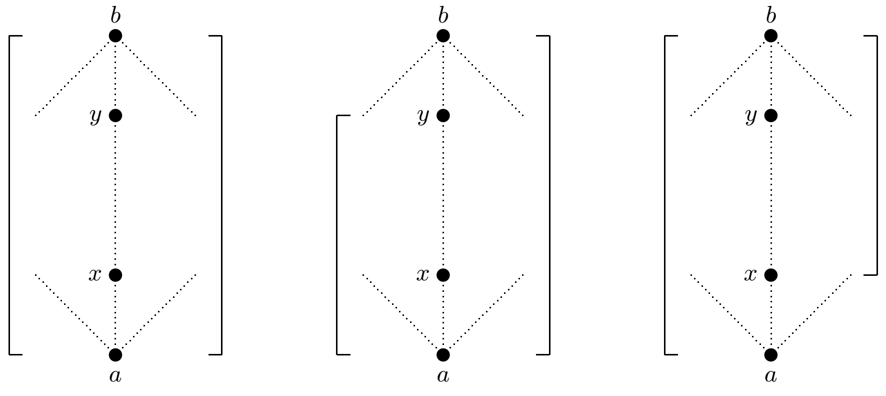
Note that if authority ranges are permitted to overlap, then immediate authority ranges are not unique. For example, suppose that `can-modify` only includes the authority ranges (ED, PL1) and (ENG1, DIR). Then both (ED, PL1) and (ENG1, DIR) are immediate authority ranges for PE1 and QE1.

A range $(x, y)$ is a *create range* if one of the following three conditions is satisfied.

- $x$ and $y$ have the same immediate authority ranges (that is, $(x) = (y)$).

- $y$ is the (upper) end point of the immediate authority range of $x$ (that is, $(x) = (a, y)$ for some $a \in R$).
- $x$ is the (lower) end point of the immediate authority range of $y$ (that is, $(y) = (x, b)$ for some $b \in R$).

The notion of a create range is perhaps easier to see pictorially, as shown in Figure 3.4. The motivation behind create ranges is that it should be possible to create roles where the parent and child roles are not the end points of an authority range, but fall within the same authority range. Note that a create range is not required to be an immediate authority range, and therefore is not necessarily an authority range.



(a) $(x) = (y) = (a, b)$ (b) $(x) = (a, y)$ (c) $(y) = (x, b)$

**Figure 3.4:** Valid conditions for a create range $(x, y)$ in RRA97: $(x)$ is indicated by the left square bracket; $(y)$ is indicated by the right square bracket

Table 3.12 shows create ranges given the authority ranges defined in Table 3.9. A tick indicates that the range is a create range. A question mark indicates the immediate authority range of at least one of the end points is not defined (see Table 3.11).

| End points | `ENG1` | `PE1` | `QE1` | `PL1` | `DIR` |
|---|---|---|---|---|---|
| `ED` | ? | ? | ? | ? | ? |
| `ENG1` | | ✓ | ✓ | ✓ | ? |
| `PE1` | | | ✓ | ✓ | ? |
| `QE1` | | ✓ | | ✓ | ? |
| `PL1` | | | | | ? |

Table 3.12: Create ranges

Finally, we can state the conditions that must be satisfied for role and edge insertions in RBAC96 according to the RRA97 model.

### Role insertion

The immediate parent and child of a new role must be the end points of a create range in the role hierarchy prior to insertion of the new role. For example, a role can be inserted between `QE1` and

`PE1`, but not between `QE1` and `DIR`. Note that the definition of role insertion implies that no role can be created which has multiple parents or children.

### Edge insertion

An edge $(x, y)$ can be inserted into the role hierarchy provided one of the following two conditions holds:

- $(x) = (y)$;
- there exists an authority range $(a, b)$ such that, either $x = a$ and $y < b$ or $x > a$ and $y = b$, and the insertion of the edge does not violate the encapsulation of $(a, b)$.

### Role deletion

RRA97 supports two alternatives for role "deletion". The first option is *role deletion*, in which $r$, the role to be deleted, is actually removed from $R$. In this case, any permission-role assignments are re-assigned to the upper shadow of $r$, and user-role assignments are re-assigned to the lower shadow of $r$. Formally,

$$\begin{aligned}
R &:= R \setminus r, \\
RH &:= RH \setminus \{(r, r') : r \lessdot r'\} \setminus \{(r', r) : r' \lessdot r\} \cup \{(r', r'') : r' \lessdot r \lessdot r''\}, \\
PA &:= PA \setminus \{(p, r) : p \in P(r)\} \cup \{(p, r') : p \in P(r), r \lessdot r'\}, \\
UA &:= UA \setminus \{(u, r) : u \in U(r)\} \cup \{(u, r') : u \in U(r), r' \lessdot r\}.
\end{aligned}$$

In this context RRA97 invokes operations from URA97 and PRA97, namely user-role assignment and revocation and permission-role assignment and revocation. It is not made clear in RRA97 whether an administrative role which can delete $r$ in the RRA97 framework necessarily has the right to perform the URA97 and RRA97 operations, or whether such operations have to be independently authorized (by the URA97 and PRA97 relations).

Role deletion is not permitted if $r$ is the end point of any range in any ARBAC97 relation. Hence, the second option is *role de-activation* in which case $r$ remains in the hierarchy and permission-role assignments remain valid. We have

$$\begin{aligned}
R &:= R, \\
RH &:= RH, \\
PA &:= PA, \\
UA &:= UA \setminus \{(u, r) : u \in U(r)\}.
\end{aligned}$$

However, a user $u$ cannot activate $r$ in a session, even if $r \in \downarrow R(u)$.

Role de-activation would usually be employed if $r$ was the end point of some range in an ARBAC97 relation, and therefore could not be deleted. We note that the effect of role de-activation can be achieved through role exclusion which was introduced in our discussion of separation of duty constraints.

**Edge deletion**

In RRA97 an edge between two roles $r$ and $r'$ can be deleted only if either $r \lessdot r'$ or $r' \lessdot r$. (Recall that we believe $RH$ should be the covering relation of the partial order on $R$, and hence *only* edges of the form $(r, r')$ would belong to $RH$. This is in common with the NIST model but not RBAC96.) The role-based access control implementation in Oracle permits the arbitrary insertion and deletion of (transitive) edges in the hierarchy (Koch and Loney 1997). This implies that the role hierarchy in Oracle is not necessarily a partial order (although the graph of the role hierarchy relation cannot contain cycles).

## 3.5 Other role-based access control models

We now briefly discuss the role graph model, the NIST model, the unified NIST RBAC model and OASIS. Our objective in this section is to outline these models and indicate their similarities to RBAC96. Indeed, we will indicate how the role graph model and the NIST model can be mapped into a subset of the RBAC96 family of models.

### 3.5.1 The role graph model

The role graph model (Nyanchama and Osborn 1994; Nyanchama and Osborn 1999) focuses on the role hierarchy and its representation as a role graph. Formally, the nodes of the role graph are ordered pairs, $(r, P(r))$, where $r$ is a role and $P(r)$ is the set of permissions explicitly assigned to $r$. The role graph has the following properties (Nyanchama and Osborn 1999):

- For any two roles $r$ and $r'$ such that $P(r) \subset P(r')$, there is a path from $r$ to $r'$.
- There is a single `MaxRole` and a path from every role to `MaxRole`. (In practice, `MaxRole` is unlikely to have any users assigned to it.)
- There is a single `MinRole` and a path from `MinRole` to every role. (In practice, $P($`MinRole`$)$, the set of permissions assigned to `MinRole`, may be the empty set.)
- The role graph is acylic. (If a cycle

  $$(r_1, r_2), (r_2, r_3), \ldots, (r_{i-1}, r_i), (r_i, r_{i+1}), \ldots, (r_{n-1}, r_n), (r_n, r_1)$$

  existed in the role hierarchy, then the permissions of each of $r_1, \ldots, r_n$ would all be identical by the first property listed above.)

In other words, the role graph is equivalent to the role hierarchy and the permission-role assignment relation of RBAC96, with the additional requirement that $R$ has a top and a bottom element (`MaxRole` and `MinRole`, respectively). The difference between the two models lies in the interpretation of the role hierarchy and the role graph. Specifically, RBAC96 assumes a role hierarchy is defined, and states the semantics of permission inheritance in terms of the role hierarchy. In contrast, the role graph model assumes the roles and their respective permission assignments are defined. Subset inclusion on permission assignments then induces a role graph.

Unlike RBAC96, the role graph model does not explicitly define how users are assigned to roles, and therefore how such assignments are to be administered.

Research on the role graph model has focused on graph modification algorithms. Such algorithms permit the modification of the role graph by inserting and deleting roles, edges and permissions. (The presence of `MaxRole` and `MinRole` simply make the role graph algorithms easier to implement.) Clearly, operations on roles and edges are equivalent to the administration of the role hierarchy as defined in RRA97, while operations on privileges correspond to administration of the permission-role assignment relation as defined in PRA97. We briefly discuss the two algorithms which add a role to the role graph. We refer the reader to the literature for details of the remaining algorithms (Nyanchama and Osborn 1999). All the role graph algorithms have time complexity polynomial in the number of nodes and edges in the role graph.

The role addition algorithm takes as parameters: the existing role graph $G$; a new role name $r$, a set of explicit permissions assigned to $r$, $P(r)$; a set of roles which are to cover $r$, $\nabla r$; and a set of roles which are to be covered by $r$, $\Delta r$. The algorithm adds the node $(r, P(r))$ to the role graph, and edges from that node to each element of $\nabla r$ and $\Delta r$. The algorithm also checks that the properties of the role graph are preserved. In particular, the algorithm checks that a path exists between two roles $r$ and $r'$ if, and only if, $P(r) \subset P(r')$. In other words, the algorithm will also update the permission-role assignments on each node where appropriate.

The role insertion algorithm takes as parameters $G$, $r$ and $P(r)$. This algorithm then deduces the appropriate place to insert the role by comparing $P(r)$ with the labelling (with respect to permissions) of the nodes in $G$.

The administration of the role graph is a centralized function, and therefore offers less flexibility than the ARBAC97 model. Furthermore, there is no explicit statement of how user-role assignments are to be managed.

*Conflict of interest* (or separation of duty) constraints are modelled by partitioning the role graph into sub-graphs which are mutually exclusive with respect to user-role assignment (Nyanchama and Osborn 1999). In particular, no user can be assigned to a role from each of two or more of these conflicting sub-graphs. This is equivalent to asserting that if two roles, $r$ and $r'$, form a conflict of interest, then no user can be assigned to any role in $\uparrow r \cap \uparrow r'$. (If such an assignment existed, then the user would be assigned to both $r$ and $r'$ either directly or indirectly. An immediate corollary of this is that no user can be assigned to `MaxRole` if there are any conflict of interest constraints. This is really stating the obvious, and is not mentioned in RBAC$_2$. The NIST model also mentions this kind of constraint on a role-based access control system (Gavrila and Barkley 1998).)

In short, the role graph model provides an alternative approach to the RBAC96 family of models, but offers little or no additional insight into role-based access control. Furthermore, there are some notable omissions: users are scarcely mentioned in the role graph model, nor is the administration of user-role assignments.

### 3.5.2 The NIST model

The outline of the NIST model was introduced in 1995 (Ferraiolo et al. 1995) and was extended to a formal description of the properties a role-based reference monitor should satisfy using first order predicate logic (Gavrila and Barkley 1998). It is essentially equivalent to $\text{RBAC}_3$ and lays particular emphasis on separation of duty constraints and sessions (referred to as subjects). Table 3.13 gives a summary of the consistency properties both in natural language and in (most cases) first order logic.[6] Where appropriate the terminology and notation have been changed from the original to maintain consistency with our presentation.

The role-based access control reference monitor that has been implemented at NIST (Gavrila and Barkley 1998) contains operations like `rmRole` (remove role) and `addAssignment` (add user-role assignment). The implementation consists of an RBAC database and an Admin Tool. The role hierarchy, user-role and permission-role assignments are stored in the RBAC database. The RBAC database also holds two binary, symmetric, irreflexive relations, `ssd` and `dsd`, which define static and dynamic separation of duty requirements.[7] In the NIST model, every role has a *cardinality*, which is defined to be the maximum number of users that may be assigned to the role. The Admin Tool verifies that an operation will preserve the consistency of the RBAC database before committing any updates.

It is shown (by modelling the database as a deterministic finite state machine) that the operations performed by the Admin Tool preserve the consistency of the RBAC database (Gavrila and Barkley 1998).

The administrative functionality of the model is centralized and less sophisticated than that of ARBAC97. Indeed, it is hard to see how some of the administrative functions can be used. For example, the `rmRole` operation requires that the role to be removed is not part of the role hierarchy, that no users are assigned to it and that it is not in a separation of duty relationship with any other role.

NIST has also implemented a role-based access control mechanism for the Web (Ferraiolo and Barkley 1997; Ferraiolo et al. 1999), called RBAC/Web, which has been developed for use with UNIX and Windows NT based servers. In its original form RBAC/Web supported a single administrative role, `admin`, but was subsequently extended using the URA97 model (Sandhu and Park 1998).

### 3.5.3 A unified role-based access control model

The NIST RBAC model is a recent attempt to develop a unified framework for role-based access control (Ferraiolo et al. 2001). Co-written by Sandhu, Ferraiolo and Kuhn, it is a synthesis of ideas in RBAC96 (Sandhu) and the work at NIST (Ferraiolo and Kuhn). The framework is arranged in four levels of increasing complexity: *flat* RBAC, *hierarchical* RBAC, *constrained* RBAC and *symmetric* RBAC. The first three of these correspond to $\text{RBAC}_0$, $\text{RBAC}_1$ and $\text{RBAC}_3$, respectively. There is an explicit requirement in flat RBAC (and hence in the higher levels) that user-role review should be supported. The fourth level introduces the requirement for permission-role review. Constrained RBAC introduces separation of duty constraints (but not cardinality constraints). There is currently no administrative component in the NIST RBAC model.

[6]Some properties have been omitted from the table, but are discussed below.

[7]In the NIST model, the symmetric and irreflexive properties of `ssd` and `dsd` are four of the consistency properties.

| | |
|---|---|
| The number of users assigned to a role does not exceed the cardinality of the role | |
| The role hierarchy contains no cycles | |
| No two roles assigned to a user form a chain | $\forall u \in U : R(u) \in \mathcal{A}(R)$ |
| No two roles assigned to a user form a static separation of duty requirement | $\forall u \in U : r_1, r_2 \in R(u) \Rightarrow (r_1, r_2) \notin \texttt{ssd}$ |
| No two roles in a user's set of sessions form a dynamic separation of duty requirement | $\forall u \in U : r_1, r_2 \in \bigcup_{s \in S(u)} R(s) \Rightarrow (r_1, r_2) \notin \texttt{dsd}$ |
| No two roles which form a static separation of duty requirement form a chain | $\forall (r_1, r_2) \in \texttt{ssd} : \{r_1, r_2\} \in \mathcal{A}(R)$ |
| No two roles which form a dynamic separation of duty requirement form a chain | $\forall (r_1, r_2) \in \texttt{dsd} : \{r_1, r_2\} \in \mathcal{A}(R)$ |
| No two roles which form a static separation of duty requirement have a common senior role | $\forall (r_1, r_2) \in \texttt{ssd} : \uparrow r_1 \cap \uparrow r_2 = \emptyset$ |
| No two roles which form a dynamic separation of duty requirement have a common senior role | $\forall (r_1, r_2) \in \texttt{dsd} : \uparrow r_1 \cap \uparrow r_2 = \emptyset$ |
| Given two roles which form a static separation of duty requirement, any role senior to one of these roles is also in a static separation of duty requirement with the other role | $\forall (r_1, r_2) \in \texttt{ssd}, \forall r > r_1 : (r, r_2) \in \texttt{ssd}$ |
| Given two roles which form a dynamic separation of duty requirement, any role senior to one of these roles is also in a dynamic separation of duty requirement with the other role | $\forall (r_1, r_2) \in \texttt{dsd}, \forall r > r_1 : (r, r_2) \in \texttt{dsd}$ |

Table 3.13: NIST consistency properties

### 3.5.4 OASIS

An interesting alternative to the role-based access control models developed in America is the OASIS (Open Architecture for Secure Interworking Services) model (Hayton 1996; Hayton et al. 1998; Hine et al. 2000; Yao et al. 2001). Unlike RBAC96, OASIS does not use a role hierarchy or a user-role assignment relation. That is, a role is essentially a collection of permissions (or capability list). Unlike RBAC96, role activation is based on the credentials in the form of roles that a user currently has active. Specifically, a user $u$ can activate a role $r$ if $u$ satisfies a *role activation rule* for $r$. A role activation rule is a Horn clause in first-order logic. That is, a role activation rule is a conjunction of constraints forming the body of the clause, which if satisfied allow the user to be assigned to the role in the head of the clause. A constraint may require that a user be currently assigned to another role or that certain temporal or environmental conditions be satisfied.

Role activation rules are similar to constraints in the URA97 and PRA97 relations in that they define certain prerequisites that a user must satisfy before he can be assigned to a role. There is less flexibility in OASIS because of the restriction that role activation rules be Horn clauses. In particular, constraints of the form $\overline{r}$ are not permitted and hence conflict of interest constraints cannot be expressed using role activation rules.

However, OASIS offers some significant advantages over the RBAC96/ARBAC97 model. Firstly, a role activation rule can insist that certain constraints be satisfied while that role is assigned to a user. This means that fine-grained event-based dynamic revocation of roles is possible in a distributed OASIS system. Secondly, de-centralized autonomous administration of role assignment can be performed by services in a distributed system using role activation rules. An overview of OASIS and its integration into a distributed system can be found in Bacon et al. (2000).

## 3.6 Computational complexity of role-based access control

We first briefly outline the depth-first search algorithm `dfs` for directed graphs (Aho and Ullman 1992). Given a directed graph $G = (V, E)$, the recursive algorithm `dfs` takes a vertex, $v$, as a parameter and traverses the graph beginning at $v$. Informally, `dfs`$(v)$ works by travelling as far as possible from $v$, marking vertices which it visits and recursively calling `dfs` for vertices which have not been visited. For example, using the role hierarchy in Figure 3.2a, the call `dfs(ENG1)` would encounter `PE1`, `PL1`, `DIR` and `QE1` (in that order). (This assumes that the data structure storing the role graph holds roles in ascending alphabetical order.) In short, `dfs`$(r)$ is equivalent to the computation of $\uparrow r$. Given a directed graph $G = (V, E)$, the running time for `dfs`$(v)$ is $O(|E|)$ (Aho and Ullman 1992).

We now consider the computational complexity of several decision problems in the RBAC96/ARBAC97 model. We denote a particular user by $u$, a particular permission by $p$, a particular role (unless otherwise stated) by $r$, and a particular administrative role by $a$. We will denote the width of $R$ by $w$ and assume that $R(u) \in \mathcal{A}(R)$. For simplicity, we will assume that we store the role graph in the *adjacency lists* representation (Aho and Ullman 1992), and that we can compute $\downarrow r$ as efficiently as $\uparrow r$. (The latter assumption may mean in practice that the transitive reduction of both $\langle R, \leqslant \rangle$ and $\langle R, \geqslant \rangle$ will need to be stored.)

$DP1$ *Should a request by u to exercise p be granted?*

In other words, is $R(p) \cap \downarrow R(u)$ non-empty? As observed above, the time taken to compute the down set for a particular role is linear in the number of edges in the role graph. Since $R(u) \in \mathcal{A}(R)$, $u$ can be assigned at most $w$ roles and we can compute $\downarrow R(u)$ in $\mathcal{O}(w|E|)$. Similarly $p$ can be assigned to at most $w$ roles. Hence $DP1$ is $\mathcal{O}(w^2|E|)$.[8]

$DP2$ *Does u (or p) satisfy a given constraint?*

We define a *primitive constraint* to be $r$ or $\overline{r}$. For a primitive constraint, $DP2$ is $\mathcal{O}(w|E|)$, since it amounts to computing for each $r' \in R(u)$ either $\downarrow r'$ or $\uparrow r'$ and checking whether

[8]In this context, note that $E = RH$.

that set contains $r$. A constraint is an arbitrary conjunction of primitive constraints.[9] Any constraint can be expressed as the conjunction of no more than $2w$ primitive constraints.[10] Hence, in general, $DP2$ is $\mathcal{O}(w^2|E|)$.

$DP3$ *Does any user (or permission) satisfy a given constraint?*

The time complexity of $DP3$ is equivalent to computing $DP2$ for all users and hence is $\mathcal{O}(w^2|E||U|)$.

$DP4$ *Can $u$ (or $p$) be assigned to $r$ by $a$ (at this moment in time)?*

In other words, does there exist a tuple $(a, c, R') \in$ `can-assign` such that $u$ satisfies $c$ and $r \in R'$? (We assume that the time taken to find a tuple of the form $(a, c, R') \in$ `can-assign` is constant.) Checking whether $r \in R' = (x, y)$ is $\mathcal{O}(|E|)$ since it is equivalent to computing $\uparrow x$ and $\downarrow y$ and checking that $r$ occurs in both. Hence, using the result for $DP2$ and assuming the number of tuples in `can-assign` is bounded by a constant independent of the sets under consideration, $DP4$ is $\mathcal{O}(|E|^2w^2)$.

$DP5$ *Can any user (or permission) be assigned to a given role by $a$ at this moment in time?*

Clearly, this is similar to the previous problem and has complexity $\mathcal{O}(|E|^2w^2|U|)$.

$DP6$ *Is the range $(x, y)$ encapsulated?*

If only the end points of the range are given, to compute the elements of $(x, y)$ we need to compute the intersection of $\uparrow x$ and $\downarrow y$ which is $\mathcal{O}(2|E| + |R|^2)$. Using Proposition 3.4.1 we can see that it is necessary to compute $\uparrow(x, y)$, $\downarrow(x, y)$, $\uparrow y$ and $\downarrow x$. Note that $\uparrow(x, y) = \uparrow x \setminus \{x\}$ and $\downarrow(x, y) = \downarrow y \setminus \{y\}$. Hence $DP6$ is $\mathcal{O}(\max\left\{4|E|, |R|^2\right\})$.

$DP7$ *Is it possible for $u$ to be assigned to $p$ (at some time in the future)?*

By this we mean "at some point in the future will there be a role to which both $u$ and $p$ are assigned?". This question can be regarded as the safety problem for the role-based access control model. The safety problem has been studied in the context of several different access control models – notably the protection matrix model (Harrison and Ruzzo 1978; Harrison et al. 1976), the take-grant model (Lipton and Snyder 1977), the schematic protection model (Sandhu 1992d) and the typed access matrix model (Sandhu 1992c). We do not know of any attempts to analyze the safety problem in role-based access control. There are two similar decision problems:

- *Is it possible for a given permission to be assigned to a given role?*
- *Is it possible for a given user to be assigned to a given role?*

In Chapter 5 we show that $DP7$ and its two associated problems are undecidable.

[9]Note that we have only considered constraints that are a conjunction of primitive constraints. However, any URA97 constraint can be written in disjunctive normal form. Hence, a tuple in `can-assign` that contains a constraint in disjunctive normal form can be replaced by several tuples in which the constraint is a conjunction of primitive constraints.

[10]Let $A \in \mathcal{A}(R)$ such that $|A| = w$. The most complicated condition that a constraint can specify is that $R(u) = A$. In our running example this would be expressed as $\texttt{PE1} \wedge \texttt{QE1} \wedge \texttt{PE2} \wedge \texttt{QE2} \wedge \overline{\texttt{PL1}} \wedge \overline{\texttt{PL2}}$. Clearly the number of primitive constraints is no more than $2w$.

*DP*8 *Is the set of constraints in* `can-assign` *consistent?*

In other words, is it possible that one entry in the `can-assign` relation allows the assignment of a user to a role whilst another entry prohibits the same assignment? (A similar question can be posed for `can-assignp`.) For example, consider the following two rows in the `can-assign` relation.

| | | |
|---|---|---|
| DSO | $\mathtt{ED} \wedge \overline{\mathtt{PL1}}$ | [PL2, PL2] |
| PSO2 | ED | (ED, PL2] |

It is clear that the first tuple prohibits the assignment of $u$ to PL2 if $\mathtt{PL1} \in R(u)$, while the second permits it.

An algorithm for deciding *DP*8 is shown in Figure 3.5. $(a_1, c_1, R_1)$ is the first tuple in the `can-assign` relation and `satisfies`$(c, u)$ is a function that returns `true` if $u$ satisfies $c$ and `false` otherwise. From Figure 3.5 we can deduce that the time complexity of *DP*8 is

$$\mathcal{O}(|U||R|(w^2|E| + |R||\mathtt{can\text{-}assign}||R|w^2|E|)) = \mathcal{O}(|U||R|^3 w^2|E||\mathtt{can\text{-}assign}|).$$

```
boolean consistent()
{
    boolean can;
    for each $(u,r) \in U \times R$                               /* $\mathcal{O}(|U||R|)$ */
    {
        can = satisfies($c_1, u$);                        /* $\mathcal{O}(w^2|E|)$ by $DP2$ */
        for each $r \in R_1$                                          /* $\mathcal{O}(|R|)$ */
        {
            for each tuple $(a, c, R') \in$ can-assign     /* $\mathcal{O}(|$can-assign$|)$ */
            {
                if $r \in R'$ and can $\neq$ satisfies($c, u$)     /* $\mathcal{O}(|R|w^2|E|)$ */
                    return false;
            }
        }
    return true;
    }
}
```

**Figure 3.5:** An algorithm for deciding whether the set of constraints in `can-assign` is consistent

## 3.7 Conclusion

In this chapter we have outlined the current state of research in role-based access control models, and presented a detailed account of the RBAC96 and ARBAC97 models. We have shown that the RBAC$_3$ model is as expressive as the role graph model and the NIST model. In the remainder of this thesis we will therefore focus on the RBAC96 and ARBAC97 models. We have briefly discussed a number of problems with these models. In the next chapter we will consider several of these issues in more detail and present some solutions. The main contributions are a family of models for administration of the role hierarchy and a complete model for administration of role-based access control. In Chapter 5 we consider the safety problem in RBAC96/ARBAC97 and show that it is undecidable in general. Chapter 7 presents a very simple framework for separation of duty policies, enabling us to consider in some detail the complexity of implementing such policies. Finally, in Chapter 8 we present a new model for role-based access control which provides, amongst other things, an alternative and more secure way to assign users and permissions to roles.

Chapter 4

# Extensions to Role-based Access Control

Role-based access control in general, and the RBAC96/ARBAC97 model in particular, provide an access control paradigm that offers a useful alternative to existing models. However, we believe there are some improvements that could be made. In Chapter 3 we have already noted certain inconsistencies, omissions and inaccuracies in RBAC96/ARBAC97. In this chapter we present a more detailed analysis of some of these issues and introduce some alternative approaches.

In the next section we consider some of the disadvantages of the RBAC96/ARBAC97 model which provide the motivation for the material in the remainder of the chapter. In Sections 4.2 and 4.3 we present the main contribution of this chapter – a new administrative model for role-based access control. In Section 4.2 we define the *administrative scope* of a role and show that this provides a natural framework for the administration of the role hierarchy. In the remainder of this section we introduce a family of increasingly sophisticated models for controlling role hierarchy operations based on the notion of administrative scope. $\text{RHA}_4$, the most complex of these models, is then compared with RRA97. In Section 4.3 we extend $\text{RHA}_4$ to SARBAC, a complete model for administration in role-based access control, and in the following section we show how SARBAC can be used to support discretionary access control in a role-based framework. In Section 4.5 we develop the notions of *ability* and *group* in role-based access control. These ideas were mentioned in passing by Sandhu and Munawer (1998b). We then propose a more general role-based access control model based on these ideas.

## 4.1 Motivation

In Section 4.1.1 we consider the overloading of the role hierarchy and two attempts at addressing the issue. In the following section we consider some of the consequences of the ARBAC97 model. In particular, we comment on the restrictions that are imposed on the structure of the role hierarchy by RRA97 and the limitations this model imposes on updates to the hierarchy.

### 4.1.1 The role hierarchy

We noted in Section 3.1 that the role hierarchy serves two purposes. Firstly, it defines the (implicit) permissions available to a role. This is referred to in Sandhu (1998) as the (*permission*) *usage* or (*permission*) *inheritance* hierarchy. The usage hierarchy determines the permissions available to a session $s$ by considering the implicit permissions of the roles in $R(s)$.

Secondly, assignment to a role $r$ enables a user to activate any role in $\downarrow r$. This is referred to as the (*role*) *activation* hierarchy. The activation hierarchy determines which roles are available to a user when initiating a session.

For example, we can see the equivalence of these two interpretations in the set of permissions assigned to a user, noted in Section 3.1, namely

$$\downarrow P(u) = \underbrace{\bigcup_{r \in R(u)} \downarrow P(r)}_{\text{usage}} = \underbrace{\bigcup_{r \in \downarrow R(u)} P(r)}_{\text{activation}}.$$

Goh and Baldwin (1998) and Lupu et al. (1995) observed that there may be situations where this duality of function is not appropriate. We now briefly discuss two important issues which are relevant to this problem.

Firstly, RBAC96 is an access control model, but we believe this aspect of the model does not accurately reflect the access control requirements of most enterprises. In Figure 3.2, is it appropriate, for example, that `DIR` has the permissions of `PE1`? Any user assigned to the role `DIR` presumably has little or no day-to-day responsibility for, or competency to perform, the activities expected of a production engineer. These issues are considered in some detail by Goh and Baldwin (1998) in their discussion of *subsidiarity*.

Secondly, the inheritance of permissions through a role hierarchy may well conflict with separation of duty constraints present in the organization. In particular, if two roles, $r_1, r_2$, form a static separation of duty requirement, then no user can be assigned to any role in $\uparrow r_1 \cap \uparrow r_2$; and if $r_1, r_2$ form a dynamic separation of duty requirement, then no user can activate any role in $\uparrow r_1 \cap \uparrow r_2$. The RBAC$_2$ model provides for the introduction of constraints, but it seems an unnecessary overhead to introduce constraints because of a feature of the model, rather than a feature of the enterprise that is the subject of the model.

A solution was proposed by Sandhu et al. (1996) that involves partial inheritance from junior roles by splitting the junior role into a "private" role (from which permissions could not be inherited) and a "normal" role. This solution, however, only addresses the issue of inheritance and not separation of duty. Furthermore, a side effect of private roles is that any range containing a private role cannot be encapsulated, preventing the use of RRA97.

A second solution was presented by Sandhu (1998) which makes a distinction between the usage and activation hierarchies. It was noted that two applications of making such a distinction are the possibility of simulating mandatory access control systems using role-based access control, and facilitating the implementation of dynamic separation of duty constraints.

The resulting model is called ERBAC (Extended RBAC) which includes a set of roles $R$ and two partial orders $\leqslant$ and $\leqslant_u$, where $r \leqslant_u r'$ implies $r \leqslant r'$. The activation hierarchy $AH$ is the covering relation of $\langle R, \leqslant \rangle$. The usage hierarchy $UH$ is the covering relation of $\langle R, \leqslant_u \rangle$. In other words, although a role $r$ has a set of roles $\{r_1, \ldots, r_k\}$ which can be activated by a user assigned to $r$, the set of permissions inherited by $r$ is a subset of the union of the explicit permissions of $r_1, \ldots, r_k$. Formally, in the context of a session $s$, we have

$$\begin{aligned} P(s) &= \bigcup_{r \in R(s)} \downarrow P(r) \\ &= \bigcup_{r \in R(s)} \{p \in P : (p, r') \in PA, r' \leqslant_u r\} \end{aligned}$$

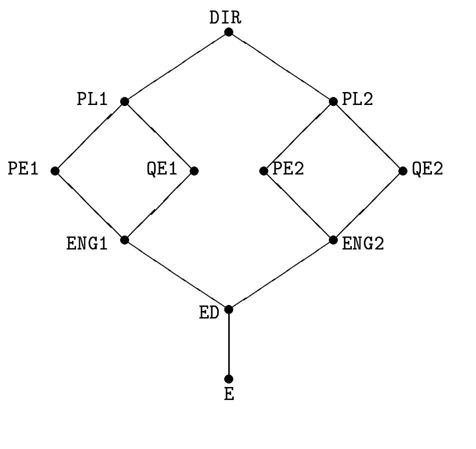
and since $r' \leqslant_u r$ implies $r' \leqslant r$

$$\begin{aligned} &\subseteq \bigcup_{r \in R(s)} \{p \in P : (p, r') \in PA, r' \leqslant r\} \\ &= \bigcup_{r \in \downarrow R(s)} \{p \in P : (p, r) \in PA\} \\ &= \bigcup_{r \in \downarrow R(s)} P(r). \end{aligned}$$

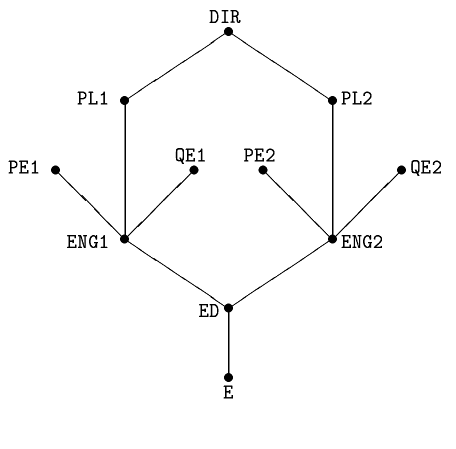
For a full account of these developments the reader is referred to Sandhu (1998).

We conclude with a brief example of the use of ERBAC hierarchies. Let us suppose that no user should have the permissions of both a Production Engineer and a Quality Engineer available in the course of any session. (That is, the roles `PE1` and `QE1`, for example, are in dynamic separation of duty.) In order to achieve this we could define the ERBAC hierarchies as depicted by the Hasse diagrams in Figure 4.1.

We observe that a user assigned to the role `PL1`, say, can activate either the role `QE1` or the role `PE1` but does not inherit the permissions of either. That is, the set of (implicit) permissions of `PL1` is not equal to the union of the explicit permissions of roles junior to it.



(a) Activation hierarchy



(b) Usage hierarchy

**Figure 4.1:** ERBAC hierarchies: Note that `PL1` does not inherit permissions of `PE1`, for example, but any user assigned to `PL1` can activate `PE1`

ERBAC certainly has merit and deals satisfactorily with both selective inheritance and dynamic separation of duty. However, no work has been done to estimate the additional cost of administering two hierarchies and keeping the usage hierarchy consistent with (changes to) the activation hierarchy. In particular, is a `can-modify` relation required for each hierarchy? It is also fair to say that some of the clarity and intuitive appeal of RBAC96 has been sacrificed. In Section 4.2.6 and Chapter 8 we suggest some alternatives to ERBAC.

Two recent papers have attempted to identify and categorize the applicability of several different hierarchies in the wider context of access control and especially in role-based access control (Moffett 1998; Moffett and Lupu 1999). We will consider one such hierarchy in Chapter 8 when motivating the secure hierarchical authorization framework.

### 4.1.2 Administration

We believe that it is appropriate that the administrative role hierarchy should have the properties ascribed to the role hierarchy. That is, if a junior administrative role has particular administrative permissions, then any senior administrative role would naturally have such permissions.

However, we also believe that the sub-models of ARBAC97 are unnecessarily complicated, particularly RRA97. Furthermore, RRA97 severely restricts the structure of role hierarchies, the form of authority ranges, and the types of role insertion and deletion which can be performed. In the case of role hierarchies, range encapsulation requires that authority ranges satisfy particular conditions which in turn limit the nature of the hierarchy. In particular, for RRA97 to be able to administer the whole of the role hierarchy, the role hierarchy must have a top and a bottom element, which we will denote $\top$ and $\bot$, respectively (these are equivalent to `MaxRole` and `MinRole` in the role graph model). Formally, for all $r \in R$, $r \leqslant \top$ and $r \geqslant \bot$. (However, we note that no administrative role can add a role with edges to and from either $\top$ or $\bot$ because the immediate authority range of these roles is not defined.)

Furthermore, the fact that the basic unit of administration is an encapsulated range severely limits the class of role hierarchies to which ARBAC97 is applicable. Figure 4.2a shows a role hierarchy that contains the single encapsulated range $(\mathtt{E}, \mathtt{ED})$; hence the `can-modify` relation will be of little use. The addition of a bottom element $(\bot)$ to the hierarchy, shown in Figure 4.2b, guarantees that the whole hierarchy forms an encapsulated range but does not form any other encapsulated ranges. In short, it is easy to find hierarchies that contain no encapsulated or few encapsulated ranges. Hence, the associated `can-modify` relation will be extremely limited in the administrative powers it can define.

We now consider several questions about the application of the RRA97 model: the first three are related to Example 3.4.1; the remaining questions are related to a further example in Sandhu and Munawer (1998b) which makes use of the usual hierarchy in Figure 3.2a. We use the authority ranges defined in the `can-modify` relation in Table 3.9.

$Q_1$ *Can administrative role* `DSO` *create role* `X` *with parent* `DIR` *and child* `QE1`?

From Table 3.12 we see that $(\mathtt{QE1}, \mathtt{DIR})$ is not a valid create range. Furthermore, the creation of role `X` would violate the encapsulation of $(\mathtt{ENG1}, \mathtt{PL1})$. Therefore RRA97 would not permit the creation of role `X`.

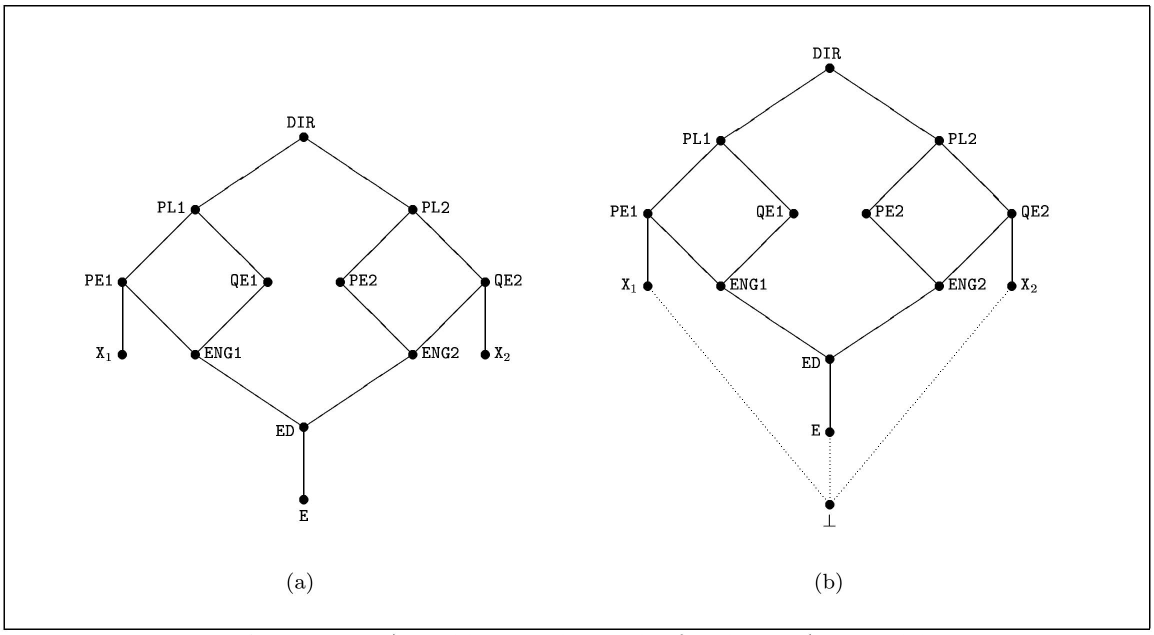


**Figure 4.2:** A problematic hierarchy for the RRA97 model

$Q_2$ *Can administrative role* PSO1 *create role* Y *with parent* PE1?

RRA97 does not permit the creation of a role with no child. Hence the role Y cannot be added to the role hierarchy by any administrative role.

$Q_3$ *Can an edge be added between* PE1 *and* QE1?

From Table 3.11 we see that $(\mathtt{PE1}) = (\mathtt{QE1}) = (\mathtt{ENG1}, \mathtt{PL1})$, and hence the edge can be added. However, it is not clear from the exposition of RRA97 which of PSO1 and DSO is authorized to make such an insertion. Clearly both end points belong to the authority range defined for PSO1 and DSO. Does the addition of this edge by DSO compromise the goals of autonomy for PSO1 and decentralization of administration? The answer is not clear.

$Q_4$ *Can administrative role* DSO *add an edge between* PE2 *and* PL1?

From Table 3.11 we have $(\mathtt{PL1}) = (\mathtt{PE2}) = (\mathtt{ED}, \mathtt{DIR})$, and hence the edge can be inserted. Note that if the can-modify relation is assumed to be symmetric with respect to the role hierarchy (that is, $(\mathtt{PSO2}, (\mathtt{ENG2}, \mathtt{PL2})) \in$ can-modify), then the edge between PL1 and PE2 cannot be inserted because neither of the two alternative criteria for edge insertion are satisfied.

$Q_5$ *Given that an edge has been added between* PE2 *and* PL1 *can administrative role* DSO *add an edge between* ENG1 *and* PE2?

If this edge were added, then $(\mathtt{ENG1}, \mathtt{PL1}) = \{\mathtt{PE1}, \mathtt{QE1}, \mathtt{PE2}\}$, and is therefore no longer an encapsulated range (since, for example, $\mathtt{PE2} \in (\mathtt{ENG1}, \mathtt{PL1})$, $\mathtt{PL2} \not\in (\mathtt{ENG1}, \mathtt{PL1})$, $\mathtt{PL2} > \mathtt{PE2}$ but $\mathtt{PL2} \not> \mathtt{PL1}$, violating (3.7)). Therefore the insertion of such an edge is not permitted.

Hence, when RRA97 is applied to our example, we have a situation where the most senior administrative role in the administrative hierarchy is unable to make certain changes to the role hierarchy because of the potential for a junior administrative role to subsequently make an undesirable role-role assignment. In short, RRA97 is rather restrictive.

We conclude this section with the following summary and miscellaneous observations about the ARBAC97 family of models.

- Further work is required on the URA97 and PRA97 models. The following comments, directed at the URA97 model, are broadly applicable to both models.
  - Revocation requires more careful consideration. As already observed, if only explicit assignments are included in the $UA$ relation, then only strong revocation makes sense. A more fine-grained approach to entitlement to activate roles might be achieved through role exclusion – the analogue of prohibited permissions.
  - As we noted in our discussion of URA97, the effect of a new user-role assignment on the set of roles already assigned to a user needs further consideration.
  - It is not clear whether a user-role assignment can only succeed if a tuple in the `can-assign` relation is satisfied. In other words, is the default behaviour in ARBAC97 that a user-role assignment fails if there is no appropriate tuple in the `can-assign` relation?
  - The definition of URA97 constraints does not need to include the disjunctive connective. For example, the tuple $(a, c_1 \vee c_2, R') \in$ `can-assign` can be replaced by the tuples $(a, c_1, R')$ and $(a, c_2, R')$.
  - URA97 does not consider whether users (acting in administrative roles) can assign roles to themselves.
  - Does URA97 apply to the assignment of administrative roles to users?
  - When determining whether to permit a user-role assignment, URA97 considers existing role assignments rather than the relevance of user characteristics. In particular, it is possible to assign a user to a role (if they satisfy an appropriate URA97 constraint) which may be far more powerful than is appropriate for the competency of the user (`dave` in Table 3.7, for example, can be assigned to the role `PL1` merely on the basis of being assigned to the role `ENG1`). In Section 7.2.2 we show how conflict of interest policies supply one solution to this problem.
  - Strictly speaking in URA97, it may not be possible to assign any role to a user $u$ if $R(u) = \emptyset$ (`fred` in Table 3.6, for example). Specifically, if there are no constraints of the form $\overline{r}$ in the `can-assign` relation, no constraint will be satisfied by $u$, and hence $u$ cannot be assigned any role by any administrative role. (The same is true of permission-role assignment.)
- RRA97 supports changes to the role hierarchy but does not permit the deletion of a role if it is the end point of a range in some ARBAC97 relation. However, the effect of other changes to the role hierarchy on URA97 and PRA97 relations is not addressed. For example, if the edge $(\texttt{PE1}, \texttt{QE1})$ is added to the role hierarchy, then the tuple $(\texttt{PSO1}, [\texttt{PL1}, \overline{\texttt{QE1}}], [\texttt{PE1}, \texttt{PE1}]) \in$ `can-assignp` can be replaced by the tuple $(\texttt{PSO1}, \texttt{PL1}, [\texttt{PE1}, \texttt{PE1}])$. (Note that the original intended semantics – that permissions held by `PE1` could not be assigned to `QE1` – have also changed, since any permission assignments will now be inherited by `QE1`.)

- In RRA97 role deletion is not permitted for roles which appear in any of the ranges of the `can-` predicates. Instead, roles are made inactive by removing all user-role and permission-role assignments. When "the references preventing deletion are suitably adjusted" the role can be deleted from the role hierarchy (Sandhu and Munawer 1998b). No mention is made of the way in which the URA97 and PRA97 relations can be modified. Presumably, certain senior administrative roles are assigned administrative permissions to update these relations, but no model is provided to constrain such updates.

- Administration of separation of duty constraints is not considered, nor is the effect of changes to the role hierarchy on role constraints. In this sense ARBAC97 should properly be called $\text{ARBAC}_1$ – the administrative model corresponding to $\text{RBAC}_1$.

- What is the relationship between administrative permissions and the ARBAC97 relations? We illustrate what we mean through the following example. Suppose $(a, c, R') \in$ `can-assign` and that $a$ tries to assign a user $u$ to a role $r$. The reference monitor must check that $u$ satisfies $c$ and that $r \in R'$. Assuming these pre-conditions are true, is it now required that $a$ has been assigned some appropriate (administrative) permission? If the answer is no, then administrative permission-role assignments are not required. If the answer is yes, then the whole administrative structure is unnecessarily complicated, because the assignment of administrative permissions can be used to dictate which RBAC96 relations an administrative role is permitted to change.

## 4.2 The RHA family of administrative models

In this section we describe a model for administration of the role hierarchy that is considerably simpler to implement and provides greater flexibility than RRA97. This is extended in Section 4.3 to a complete model for administration of role-based access control. We shall also see that our model obviates or solves several problems identified in the preceding section.

Our model is motivated by the following two intuitively reasonable suggestions for resolving the problem introduced in Example 3.4.1. Namely, once role `X` has been created:

- Remove `QE1` from `PSO1`'s administrative range as `QE1` is now less than `X`, a role which is not in `PSO1`'s administrative range. That is, only `DSO` and above should now be able to administer `QE1`. In particular, `PSO1` would not be able to make `PE1` less than `QE1`.

- A role $r$ such that $|\nabla r| > 1$ (such as `QE1` once `X` has been inserted into the hierarchy) must be administered by a role which has administrative control over every role in $\nabla r$. In our example, only `DSO` would be able to make `PE1` less than `QE1`.

These solutions have a similar approach and could be implemented by imposing upper limits on the authority of each administrative role. Therefore, we introduce the notion of administrative scope which is motivated by Proposition 3.4.1.

**Definition 4.2.1** *The* administrative scope *of $r \in R$, denoted $\sigma(r)$, is defined to be the set $\{s \in R : s \leqslant r,\ \uparrow s \setminus \uparrow r \subseteq \downarrow r\}$.*

In our usual hierarchy, $\texttt{ENG1} \in \sigma(\texttt{PL1})$ because $\uparrow\texttt{ENG1} = \{\texttt{ENG1}, \texttt{PE1}, \texttt{QE1}, \texttt{PL1}, \texttt{DIR}\}$ and $\uparrow\texttt{PL1} = \{\texttt{PL1}, \texttt{DIR}\}$; hence $\uparrow\texttt{ENG1} \setminus \uparrow\texttt{PL1} = \{\texttt{ENG1}, \texttt{PE1}, \texttt{QE1}\} \subset \downarrow\texttt{PL1}$. In fact, it can easily be verified that $\sigma(\texttt{PL1}) = \{\texttt{ENG1}, \texttt{PE1}, \texttt{QE1}, \texttt{PL1}\}$. However, $\texttt{ENG1} \notin \sigma(\texttt{PE1})$, for example, because $\texttt{QE1} \notin \downarrow\texttt{PE1}$. Table 4.1 shows the "non-trivial" administrative scope of roles in Figure 3.2a. (That is, Table 4.1 only includes $r$ if $\sigma(r) \neq \{r\}$.)

| Role | Administrative scope |
|---|---|
| DIR | $R$ |
| PL1 | $\{\texttt{ENG1}, \texttt{PE1}, \texttt{QE1}, \texttt{PL1}\}$ |
| PL2 | $\{\texttt{ENG2}, \texttt{PE2}, \texttt{QE2}, \texttt{PL2}\}$ |
| ED | $\{\texttt{E}, \texttt{ED}\}$ |

Table 4.1: Administrative scope in $\text{RHA}_1$

Informally, administrative scope has characteristics similar to those exhibited at the upper end point of an encapsulated range. That is, there is only one way into the administrative scope of $r$ from above and that is through $r$ itself. More formally, we have the following proposition which shows that administrative scope is a less restrictive notion than that of range encapsulation.

**Proposition 4.2.1** *If $(x, y)$ is an authority range, then $(x, y) \subseteq \sigma(y)$.*

**Proof** Suppose $z \in (x, y)$. Then $x < z < y$ and hence $\uparrow x \supset \uparrow z \supset \uparrow y$. Therefore,

$$\begin{aligned} \uparrow z \setminus \uparrow y &\subset \uparrow(x, y) \setminus \uparrow y \\ &= (x, y) \quad \text{by (3.9)} \\ &\subset \downarrow y. \end{aligned}$$

That is, $z \in \sigma(y)$. ■

The administrative scope of a role is determined by the role hierarchy and changes dynamically as the hierarchy changes. (This is in contrast to RRA97, where administration is determined by the `can-modify` relation, which in turn imposes restrictions on changes that can be made to the hierarchy.) For example, if `X` is added to the hierarchy, where $\texttt{QE1} \lessdot \texttt{X} \lessdot \texttt{DIR}$, then `QE1` is no longer in $\sigma(\texttt{PL1})$. Figure 4.3 shows how the administrative scope of `PL1` changes as edges and roles are added to the hierarchy.

By definition, we have that $r \in \sigma(r)$ for all $r \in R$. We therefore introduce the notion of *proper* administrative scope which is defined to be $\sigma(r) \setminus \{r\}$ and is denoted $\sigma^+(r)$. If $r' \in \sigma^+(r)$, we say $r$ is an *administrator* of $r'$.

**Proposition 4.2.2** *If $r \lessdot r'$ and $r \in \sigma^+(a)$ for some $a \in R$, then $r' \in \sigma(a)$.*

**Proof** Suppose $r' \notin \sigma(a)$. Then there exists $r'' \in \uparrow r' \setminus \uparrow a$ such that $r'' \notin \downarrow a$. That is, $r < r' \leqslant r''$ and $r'' \notin \downarrow a$. Hence, $r \notin \sigma(a)$, which is a contradiction. ■

**Proposition 4.2.3** *If $Ad(r) \neq \emptyset$, then $Ad(r)$ has a unique minimal element which we call the* line manager *of $r$.*

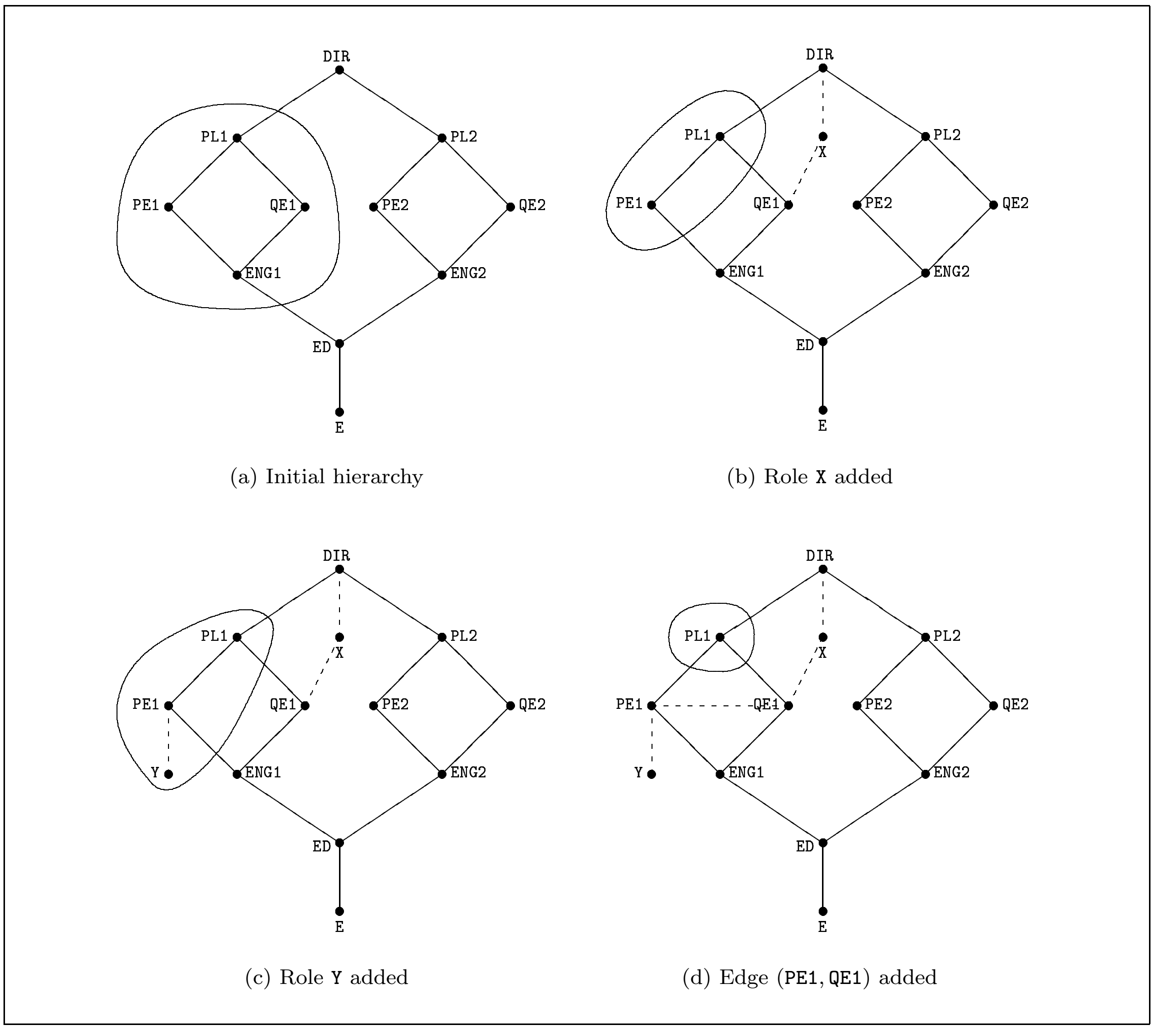


**Figure 4.3:** The dynamic nature of administrative scope: The roles inside the closed curve denote the administrative scope of PL1

**Proof** If $|Ad(r)| = 1$, then the result follows immediately. Therefore, suppose that $x \neq y$ are both minimal elements in $Ad(r)$. Then $y \in \uparrow r$ because $r \in \sigma^+(y)$ and $y \notin \uparrow x$ (since $x$ and $y$ are minimal elements and hence $x \not< y$). Furthermore, by definition, $\uparrow r \setminus \uparrow x \subseteq \downarrow x$ and hence $y \in \downarrow x$. Hence, $y < x$ and $x$ is not a minimal element in $Ad(r)$. ■

Note that the case $Ad(r) = \emptyset$ occurs when $r$ is not contained in the proper administrative scope of any other role. (An obvious example is the DIR role.)

We believe the concept of line manager can be used to support decentralization and autonomy in the administration of role-based access control. The development of this topic is beyond the scope of the thesis, although we briefly discuss the matter further in Chapter 9.

We now consider the ways in which administrative scope can be used as a foundation for administration of the role hierarchy. This gives rise to several alternative models which we present in order of increasing complexity. For convenience we will label these models RHA$_i$, where RHA is an abbreviation for role hierarchy administration. In Section 4.3 we extend RHA$_4$ to SARBAC, a complete model for administration in role-based access control.

The development of our model is based on the assumption that changes to the hierarchy can only be made by a role whose administrative scope includes all roles affected by the changes. For example, once `X` has been added only `DIR` would be permitted to make changes to the hierarchy that affect `QE1`. In particular, `DIR` can add the edge between `PE1` and `QE1`, but `PL1` cannot.

Hierarchy operations may cause additional edge insertions and deletions. In particular, insertion operations may introduce transitive edges that need to be deleted and deletion operations may require the insertion of edges that would otherwise be lost. Simple examples of this behaviour can be found in Figures 8.1 and 8.2. In order to focus on the development of our model, we postpone a detailed discussion of these issues until Chapter 8 in which we present a new model for role-based access control.

We also assume that edge and role insertions are not permitted to introduce cycles into the hierarchy. For convenience we introduce the following notation for hierarchy operations: $\texttt{AddRole}(a, r, \Delta r, \nabla r)$ means role $a$ inserts role $r$ into the hierarchy with immediate children $\Delta r \subseteq R$ and immediate parents $\nabla r \subseteq R$; $\texttt{DeleteRole}(a, r)$ means role $a$ deletes role $r$ from $R$; $\texttt{AddEdge}(a, r, r')$ means role $a$ adds $(r, r')$ to $RH$; $\texttt{DeleteEdge}(a, r, r')$ means role $a$ deletes $(r, r')$ from $RH$. Then

$$\texttt{AddRole}(a, r, \Delta r, \nabla r) \text{ succeeds provided } \Delta r \subseteq \sigma^+(a) \text{ and } \nabla r \subseteq \sigma(a); \tag{4.1}$$

$$\texttt{DeleteRole}(a, r) \text{ succeeds provided } r \in \sigma^+(a); \tag{4.2}$$

$$\texttt{AddEdge}(a, r, r') \text{ succeeds provided } r, r' \in \sigma(a); \tag{4.3}$$

$$\texttt{DeleteEdge}(a, r, r') \text{ succeeds provided } r \in \sigma^+(a). \tag{4.4}$$

Clearly these are far simpler requirements (both to state and implement) than those imposed on hierarchy operations in RRA97. Note that we do not require that a new role has a single parent and child, unlike in RRA97. Nor do we require that at least one of $\Delta r$ and $\nabla r$ be non-empty. Notice that by Proposition 4.2.2 we do not need to check that $\nabla r \subseteq \sigma(a)$ when deleting role $r$, nor that $r' \in \sigma(a)$ when deleting the edge $(r, r')$.

### 4.2.1 RHA$_1$ – The base model

In this model we make no distinction between roles and administrative roles. In RHA$_1$ a hierarchy operation succeeds provided one of the conditions (4.1) – (4.4) is satisfied. We note that such an approach is unlikely to meet the security requirements of an organization. In particular, it is probably not desirable for roles near the bottom of the hierarchy to have any administrative power. For example, in Table 4.1, `E` is in the administrative scope of `ED`, but it is unlikely that `ED` should have any control over the hierarchy.

However, RHA$_1$ does have the advantage of great simplicity. Furthermore, RHA$_1$ can be implemented without introducing any additional relations, unlike ARBAC97.

### 4.2.2 RHA$_2$ – Administrative permissions

In this model, like in RHA$_1$, we make no distinction between roles and administrative roles. We do however introduce administrative permissions. In RHA$_2$ $a$ can perform a hierarchy operation provided one of the conditions (4.1) – (4.4) is satisfied and $a$ has the appropriate administrative permissions. (Clearly RHA$_1$ is a special case of RHA$_2$ in which the set of administrative permissions is empty.)

RHA$_2$ can be implemented without introducing additional relations and offers finer granularity than RHA$_1$ without incurring any significant overheads. We can envisage, for example, `DIR` having (administrative) permissions which enable the role to make changes to the hierarchy and to the assignment relations, while `PL1` and `PL2` only have permissions to change assignment relations, and `ED` has no administrative permissions.

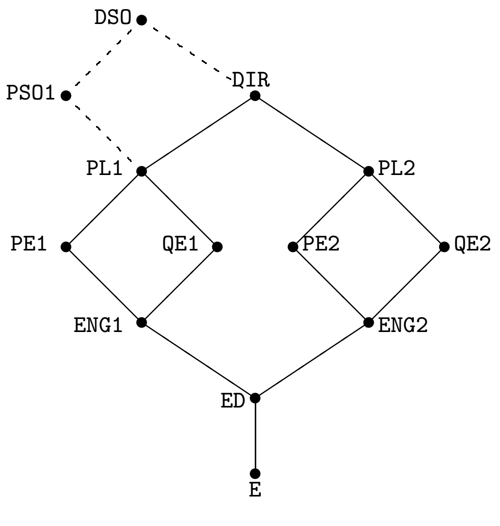
### 4.2.3 RHA$_3$ – The `admin-authority` relation

In this model we introduce a binary relation `admin-authority` $\subseteq R \times R$. If $(a, r) \in$ `admin-authority` we say $a$ is an *administrative role* and also that $a$ *controls* $r$. We denote the set of roles that $a$ controls by $C(a)$.

We first make the observation that the `admin-authority` induces an *extended* hierarchy on the set of roles which includes the original hierarchy. For example, the `admin-authority` relation defined in Figure 4.4a results in the extended hierarchy in Figure 4.4b. The elements of `admin-authority` are represented by broken lines. (In the remainder of the thesis we will sometimes visualize examples using an extended hierarchy rather than explicitly defining the `admin-authority` relation.)

| `admin-authority` | |
|---|---|
| Administrative Role | Role |
| PSO1 | PL1 |
| DSO | DIR |
| DSO | PSO1 |

(a) The `admin-authority` relation



(b) The extended hierarchy

**Figure 4.4:** An extended hierarchy

We extend the definition of administrative scope in a natural way: namely,

$$\sigma(a) = \{r \in R : \uparrow r \setminus \uparrow C(a) \subseteq \downarrow C(a)\} \quad \text{and} \quad \sigma^+(a) = \sigma(a) \setminus C(a),$$

where the evaluation of $\uparrow r$, $\uparrow C(a)$ and $\downarrow C(a)$ takes place in the extended hierarchy. For example, in Figure 4.4, $\sigma(\texttt{PSO1}) = \{\texttt{ENG1}, \texttt{PE1}, \texttt{QE1}, \texttt{PL1}\}$ and $\sigma(\texttt{DSO}) = \{\texttt{E}, \ldots, \texttt{DIR}, \texttt{PSO1}\}$.

An administrative role can perform a hierarchy operation provided one of the conditions (4.1) – (4.4) is satisfied. There are two self-evident consistency requirements that `admin-authority` must satisfy:

$$\text{for all } (a, r) \in \texttt{admin-authority},\ a \not< r; \tag{4.5}$$

$$(a, r) \in \texttt{admin-authority} \text{ implies } (r, a) \notin \texttt{admin-authority}. \tag{4.6}$$

RHA$_3$ provides a level of indirection not available in RHA$_1$ and RHA$_2$, and therefore can be used to implement a far more flexible security policy stating which (administrative) roles have responsibility for which parts of the role hierarchy. In this sense the `admin-authority` relation is similar to the `can-modify` relation in RRA97. For example, the `can-modify` relation in Table 3.9 can be replaced by the `admin-authority` relation in Figure 4.4a. (The tuple $(\texttt{DSO}, \texttt{PSO1})$ is included for the development and discussion of the model in Section 4.2.4.)

Clearly, RHA$_3$ offers greater flexibility than RHA$_1$ and RHA$_2$. The overheads incurred as a result of this flexibility are not significant and are considerably less than those incurred by the use of the `can-modify` relation in RRA97. We also note that RHA$_1$ is a special case of RHA$_3$, where for all $r \in R$, $(r, r) \in \texttt{admin-authority}$.

## 4.2.4 RHA$_4$ – Administering the `admin-authority` relation

In this section we consider how RHA$_3$ can be extended to administer the `admin-authority` relation. We need to consider when and how the `admin-authority` relation can be updated by hierarchy operations and by the actions of administrative roles.

### Updates by administrative roles

Removing an element from `admin-authority` corresponds to removing an edge from the extended hierarchy. Therefore, $(a, r)$ can be removed from `admin-authority` by role $a'$ provided $r \in \sigma^+(a')$ and $a \in \sigma(a')$. If $r$ is removed from $\sigma(a')$ as a result of deleting $(a, r)$, then it is necessary to add $(a', r)$ to `admin-authority` in order to preserve the administrative scope of $a'$. For example, given the `admin-authority` relation in Figure 4.4a, `DSO` can remove $(\texttt{PSO1}, \texttt{PL1})$ from the relation. In this case it is not necessary to add $(\texttt{DSO}, \texttt{PL1})$ to `admin-authority` since $(\texttt{DSO}, \texttt{DIR}) \in \texttt{admin-authority}$ and hence $\texttt{PL1} \in \sigma(\texttt{DIR})$. However, if `DSO` removes $(\texttt{PSO1}, \texttt{ENG1})$ from the extended hierarchy depicted in Figure 4.5d, then $(\texttt{DSO}, \texttt{ENG1})$ must be added to the `admin-authority` relation.

Similarly, $(a, r)$ can be added to `admin-authority` by role $a'$ provided $r \in \sigma^+(a')$ and $a \in \sigma(a')$. This may require the deletion of "transitive edges" from `admin-authority`. For example, if we perform the operation $\texttt{AddRole}(\texttt{DSO}, \texttt{PL1}', \{\texttt{PE1}, \texttt{QE1}\}, \emptyset)$ on the hierarchy in Figure 4.5b followed by the addition of $(\texttt{PSO1}, \texttt{PL1}')$ to `admin-authority` then we must remove $(\texttt{PSO1}, \texttt{PE1})$ and $(\texttt{PSO1}, \texttt{QE1})$ from `admin-authority`.

**Updates by hierarchy operations**

We need to define the behaviour of our model for the following extended hierarchy operations: $\texttt{AddRole}(a, r, \Delta r, \emptyset)$, $\texttt{DeleteRole}(a, r)$, $\texttt{AddEdge}(a, r, r')$ and $\texttt{DeleteEdge}(a, r, r')$.

Figure 4.5, based on the hierarchy and `admin-authority` relation in Figure 4.4, shows an example of each of the above situations and provides a schematic motivation for the behaviour of the model. We assume that edges implied by transitivity, which would be lost as a result of a hierarchy operation, are made explicit following the operation. An example of this is the addition of the edge $(\texttt{ED}, \texttt{PE1})$ in Figure 4.5d following the deletion of the edge $(\texttt{ENG1}, \texttt{PE1})$.

$\texttt{AddRole}(a, r, \Delta r, \emptyset)$ In Figure 4.5a we see that it is necessary to connect the new role `X` to the extended hierarchy. The obvious way to do this is to add $(\texttt{PSO1}, \texttt{X})$ to the `admin-authority` relation. Hence, the operation $\texttt{AddRole}(a, r, \Delta r, \emptyset)$ requires that $(a, r)$ be added to the `admin-authority` relation.

$\texttt{DeleteRole}(a, r)$ In Figure 4.5b we see that if $(a', r) \in \texttt{admin-authority}$, then we must re-connect $a'$ to the extended hierarchy. The obvious way to do this is to add $(a', r')$ to `admin-authority` for all $r' \in \Delta r \cap \sigma(a')$. In Figure 4.5b, we add $(\texttt{PSO1}, \texttt{PE1})$ and $(\texttt{PSO1}, \texttt{QE1})$ to `admin-authority`.

$\texttt{AddEdge}(a, r, r')$ In Figure 4.5c we see that if $(a, r)$ and $(a, r')$ belong to `admin-authority` then we can remove $(a, r)$ from `admin-authority` because $(a, r)$ has become a transitive edge in the extended hierarchy. In Figure 4.5c we remove $(\texttt{PSO1}, \texttt{PE1})$ from `admin-authority`.

$\texttt{DeleteEdge}(a, r, r')$ Finally, in Figure 4.5d we see that we may have to re-connect $r$ to the extended hierarchy by inserting pairs into `admin-authority` that correspond to transitive edges in the extended hierarchy. In Figure 4.5d we insert $(\texttt{PSO1}, \texttt{ENG1})$ into `admin-authority`.

**Antichains in RHA$_4$**

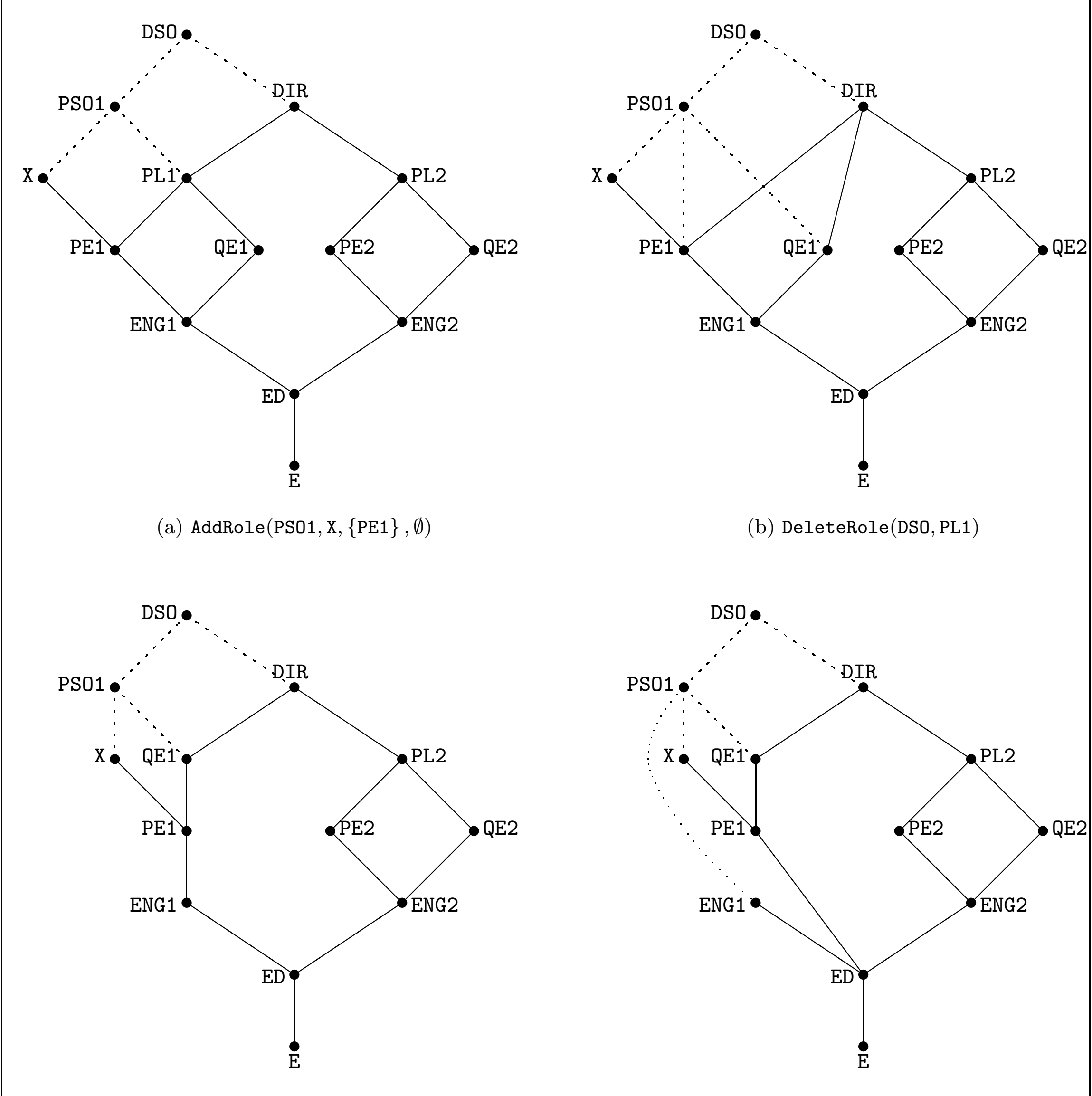
Superficially it would seem attractive (and highly appropriate in the context of the thesis) to require that $C(a)$ be an antichain. However, Figure 4.5b provides an example of why such a requirement is unsuitable. Namely, when `PL1` is deleted, both `X` and `PE1` should belong to $C(\texttt{PSO1})$ despite the fact that $\texttt{PE1} \lessdot \texttt{X}$.

### 4.2.5 A comparison of RHA$_4$ and RRA97

We now examine the relative merits of RHA$_4$ and RRA97 by reconsidering the questions we posed earlier in this section. We base our comparison on the `can-modify` and `admin-authority` relations given in Table 3.9 and Figure 4.4a, respectively.

Q$_1$ *Can administrative role* `DSO` *create role* `X` *with parent* `DIR` *and child* `QE1`?

Since $\texttt{QE1}, \texttt{DIR} \in \sigma(\texttt{DSO})$, the answer to Q$_1$ is "yes", unlike in RRA97, in which the addition of such an edge violates range encapsulation. Note that `QE1` is no longer in the administrative scope of `PSO1` (see Figure 4.3b). In other words, role additions are more likely to be permitted, although the administrative scope of roles may change.



(a) AddRole(PSO1, X, {PE1}, ∅) (b) DeleteRole(DSO, PL1)

(c) AddEdge(PSO1, PE1, QE1) (d) DeleteEdge(PSO1, ENG1, PE1)

**Figure 4.5:** Updates to the extended hierarchy

$Q_2$ *Can administrative role* PSO1 *create role* Y *with parent* PE1?

Since PE1 $\in \sigma($PSO1$)$, the answer to this question is "yes", unlike in RRA97, which does not permit the creation of a role with no child role in the hierarchy. (Even if RRA97 did permit the creation of a role with no child, the creation of Y would violate the encapsulation of [ENG1, PL1].)

$Q_3$ *Can an edge be added between* PE1 *and* QE1?

PE1, QE1 $\in \sigma($PSO1$)$, so both DSO and PSO1 can add this edge.

$Q_4$ *Can administrative role* DSO *add an edge between* PL1 *and* PE2?

As in RRA97, the answer to this question is "yes". Note that it would make no difference if $(\texttt{PSO2}, \texttt{PL2}) \in$ `admin-authority` in Figure 4.4a, unlike in RRA97. However, the administrative scope of `PSO2` is reduced to $\{\texttt{QE2}, \texttt{PL2}\}$. `PE1` and `QE1` remain in the administrative scope of `PSO1`.

$Q_5$ *Given that an edge has been added between* `PL1` *and* `PE2`, *can administrative role* `DSO` *add an edge between* `PE2` *and* `ENG1`?

Unlike in RRA97, the addition of this edge is permitted since $\texttt{PE2}, \texttt{ENG1} \in \sigma(\texttt{DSO})$.

These examples show that $\text{RHA}_4$ is less restrictive than RRA97 since the preservation of encapsulated ranges significantly limits the changes that can be made to the role hierarchy. For example, in RRA97, the addition of `X` is prohibited because it violates range encapsulation. Using $\text{RHA}_4$, the addition of `X` to the hierarchy merely changes the administrative scope of `PSO1`.

$\text{RHA}_4$ is also more widely applicable than RRA97; there is nothing to prevent $\text{RHA}_4$ being applied to the hierarchy in Figure 4.2, for example.

It is clear that the `admin-authority` relation is far simpler to implement and maintain than `can-modify`. For example, the `can-modify` relation can only contain encapsulated ranges whereas `admin-authority` contains single roles. (We note that worst case upper bounds for the number of tuples in `admin-authority` and `can-modify` are $|R|^2$ and $|AR| \cdot 2^{|R|}$ respectively.) The conditions imposed on tuples $(a, r) \in$ `admin-authority` given in (4.5) and (4.6) are relatively simple compared to the requirements of an encapsulated range.

We also note that the `admin-authority` relation and the associated extended hierarchy provides a clearer and intuitive representation of the administrative capabilities of each administrative role. In short, the `admin-authority` relation provides a more widely applicable and intellectually appealing basis for administration of the role hierarchy than the `can-modify` relation.

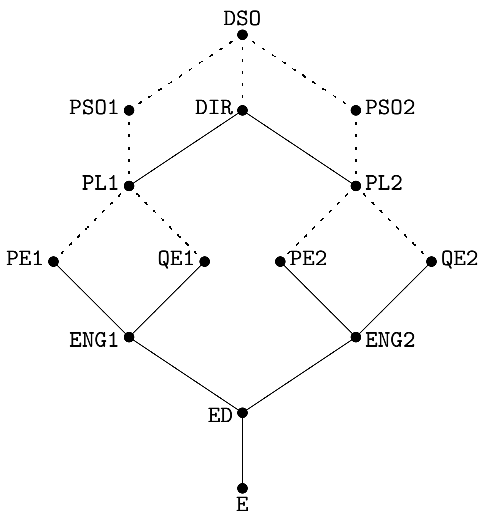
### 4.2.6 $\text{RHA}_4$ and the role hierarchy

The use of indirection in the `admin-authority` relation means we can address some of the problems identified with permission inheritance in the role hierarchy. Figure 4.6b shows an extended hierarchy based on the `admin-authority` relation in Figure 4.6a which provides an alternative to the hierarchy depicted in Figure 3.2a. Such a hierarchy cannot be administered using RRA97. Note that the definition of $\text{RHA}_4$ does not prohibit a tuple such as $(\texttt{PL1}, \texttt{PE1})$ appearing in the `admin-authority` relation. If we wish to distinguish between roles ($R$) and administrative roles ($AR$) as in RBAC96 and insist that `admin-authority` $\subseteq AR \times R$, then we can replace $(\texttt{PL1}, \texttt{PE1})$ with $(\texttt{PSO1}, \texttt{PE1})$. (This flexibility in the `admin-authority` relation will be exploited further when we consider supporting discretionary access control using roles in Section 4.4.) Note that the inheritance between the senior roles and the more functional roles has been removed.

Figure 4.6c shows revised user-role assignments based on those in Table 3.2. We see that we can now restrict the permissions of `claire`, who is assigned to the `DIR` role, to those of the senior roles. This corresponds more accurately with the deployment of responsibilities in a real-world enterprise than in the original example. Furthermore, `claire` is assigned to the `DSO` role which means that she can administer junior roles in the hierarchy although she is not actually assigned to those roles. It has been argued (Sadighi Firozabadi and Sergot 1999; Sadighi Firozabadi et al.

| admin-authority | |
|---|---|
| Administrative Role | Role |
| DSO | DIR |
| DSO | PSO1 |
| DSO | PSO2 |
| PSO1 | PL1 |
| PL1 | PE1 |
| PL1 | QE1 |
| PSO2 | PL2 |
| PL2 | PE2 |
| PL2 | QE2 |

(a)



(b)

| $u$ | $R(u)$ | $\downarrow R(u)$ |
|---|---|---|
| anne | {QE1, QE2} | {QE1, ENG1, ED, E, QE2, ENG2} |
| bill | {PE1, QE1, PL1} | {PL1, PE1, QE1, ENG1, ED, E} |
| claire | {ED, DIR, DSO} | {DSO, DIR, PL1, PL2, ED, E} |
| dave | {ENG1} | {ENG1, ED, E} |
| emma | {PE1, QE2} | {PE1, ENG1, ED, E, QE2, ENG2} |

(c)

**Figure 4.6:** Using RHA$_4$ to reduce inheritance in the role hierarchy

2001) that this distinction between having the authority to assign a permission (or role) and being assigned a permission (or role) is an important and largely ignored area in access control. Note also that it is necessary for bill to be assigned explicitly to PE1 and QE1 in order for him to make use of the permissions available to roles in the project for which he is responsible. This corresponds more accurately with the spirit of least privilege.

In other words, the use of RHA$_4$ means that the role hierarchy can have a variety of forms. Specifically, unlike in RRA97, we do not require ranges for administrative purposes, nor do we impose conditions on any ranges in the hierarchy. Another beneficial side effect is that there are fewer edges in the role hierarchy, thereby reducing the running time of the dfs algorithm (see Section 3.6). The only overhead that has been introduced is the admin-authority relation. However, the tuples in this relation and the conditions they are required to satisfy are clearly less complex than those in the can-modify relation.

## 4.3 SARBAC: An administrative model for role-based access control

We now describe how RHA$_4$ can be extended in a very simple way to provide a complete administrative model for role-based access control. We call this model SARBAC – scoped administration of role-based access control.

Our intention is that SARBAC can be used as an alternative model to ARBAC97 for administration in conjunction with RBAC96. However, we will assume that, as in the NIST model, for all $u \in U$ and all $p \in P$, $R(u), R(p) \in \mathcal{A}(R)$. We will also assume that there is no distinguished set of administrative roles or administrative permissions. In particular, an administrative role and its administrative powers (scope) are determined solely by the `admin-authority` relation.

### 4.3.1 Administration of user-role and permission-role assignment

In order to exercise more control over the assignment of roles to users and permissions, we first define a SARBAC constraint.

**Definition 4.3.1** *Let $R' = \{r_1, \dots, r_k\}$ be a subset of $R$ and let $\bigwedge R'$ denote $r_1 \wedge \cdots \wedge r_k$. Then a* SARBAC constraint *has the form $\bigwedge R'$ for some $R' \in 2^R$.*

*A constraint $\bigwedge R'$ is satisfied by $u \in U$ if for all $r \in R'$, $r \in {\downarrow}R(u)$. A constraint $\bigwedge R'$ is satisfied by $p \in P$ if for all $r \in R'$, $r \in {\uparrow}R(p)$.*

Note that $\bigwedge \emptyset$ is trivially satisfied by all users and permissions (and corresponds to the URA97 constraint $r \vee \overline{r}$). We also note that it is sufficient to define a constraint to be $\bigwedge A$, for some antichain $A$, rather than $\bigwedge R'$. In particular, we have the following result.

**Proposition 4.3.1** *Let $R' \subseteq R$. Then $\bigwedge R'$ is satisfied by a user $u$ if, and only if, $\bigwedge \overline{R'}$ is satisfied by $u$, where $\overline{R'}$ is the set of maximal elements in $R'$. Similarly, $\bigwedge R'$ is satisfied by a permission $p$ if, and only if, $\bigwedge \underline{R'}$ is satisfied by $p$, where $\underline{R'}$ is the set of minimal elements in $R'$.*

**Proof**

- $\Rightarrow$ The result follows immediately since $R' \supseteq \overline{R'}$.
- $\Leftarrow$ Suppose $u$ satisfies $\bigwedge \overline{R'}$. We can assume $\overline{R'} \subset R'$ (otherwise we are done). Hence let $r \in R' \setminus \overline{R'}$. Then there exists $r' \in \overline{R'}$ such that $r < r'$. Now $r' \in {\downarrow}R(u)$, since $u$ satisfies $\bigwedge \overline{R'}$ by assumption, and hence $r \in {\downarrow}R(u)$.

The proof for permissions is similar and is omitted. ■

We define two relations `ua-constraints` $\subseteq R \times \mathcal{A}(R)$ and `pa-constraints` $\subseteq R \times \mathcal{A}(R)$. The purpose of these relations is similar to `can-assign` and `can-assignp` in ARBAC97, in that they control whether a given user or permission can be assigned to a particular role. Specifically, an administrative role $a$ can assign a user $u$ to a role $r$ provided there exists a tuple $(r, A) \in$ `ua-constraints` such that $u$ satisfies $\bigwedge A$ and $r$ is in the administrative scope of $a$. In this case, $R(u) := \overline{R(u) \cup \{r\}}$, since $R(u)$ is required to be an antichain. Hence, if $r \leqslant r'$ for some $r' \in R(u)$, then the user-role assignment has no effect. (To simplify the presentation we assume that for all $r \in R$ there exists $(r, A) \in$ `ua-constraints` and $(r, A') \in$ `pa-constraints`. We note that in practice, it will be simpler to omit tuples of the form $(r, \emptyset)$ from `ua-constraints`; the assignment of a user $u$ to such a role $r$ succeeds provided $r$ is in the administrative scope of $a$.) The administrative role $a$ can assign a permission $p$ to a role $r$ provided there exists a tuple $(r, A) \in$ `pa-constraints` such that $p$ satisfies $\bigwedge A$ and $r$ is in the administrative scope of $a$. In this case, $R(p) := \underline{R(p) \cup \{r\}}$.

The administrative role $a$ can revoke $(u, r) \in UA$ provided $r$ is in the administrative scope of $a$. Similarly, $a$ can revoke $(p, r) \in PA$ provided $r$ is in the administrative scope of $a$. We do not support weak revocation.

Unlike in URA97 and PRA97, we do not include constraints of the form $\overline{r}$ in the `ua-constraints` and `pa-constraints` relations. There are several reasons for this. Firstly, every entry in `ua-constraints` and `pa-constraints` has the form $(r, A)$, where $A$ is an antichain. In other words, the representation has the virtue of simplicity. Secondly, we are effectively using Horn clause logic, and hence it is simple and efficient to implement. Furthermore, an inconsistent specification of constraints on user- and permission-role assignment cannot arise (unlike in ARBAC97). Thirdly, updates to the `ua-constraints` and `pa-constraints` relations will be much easier in the event of deletions and additions to the role hierarchy. (We consider this in more detail below.) Finally, we believe that separation of duty constraints are a more suitable instrument for preventing assignments of certain users to certain roles. For example, $(\mathtt{DSO}, \mathtt{ED} \wedge \overline{\mathtt{PL1}}, [\mathtt{PL2}, \mathtt{PL2}]) \in$ `can-assign` in Table 3.5 means that `PL1` and `PL2` cannot be assigned to the same user. This is merely a static separation of duty constraint. (We are assuming of course that any user-role assignment will check that no separation of duty constraints will be violated by the assignment. Note that the following situation could arise in SARBAC: $(r, \{r_1, r_2\}) \in$ `ua-constraints` with $r_1$ and $r_2$ also forming a static separation of duty constraint. In this case, no user can be assigned to $r$. Of course this situation can also arise in ARBAC97.)

We do not include disjunctions in `ua-constraints` unlike `can-assign`. We note, however, that we can express disjunctions in a similar way to alternative clauses in a Prolog program. That is, $(r, (r_1 \wedge r_2) \vee r_3)$ would be expressed as two tuples in `ua-constraints`: namely, $(r, \{r_1, r_2\})$ and $(r, \{r_3\})$. Note that a tuple in the `ua-constraints` relation is very similar to a role activation rule in OASIS.

### Updates to the `ua-constraints` relation

A role $a$ can add a tuple $(r, A)$ to (or delete a tuple from) `ua-constraints` or `pa-constraints` provided $r \in \sigma(a)$ and $A \subseteq \sigma(a)$. We now consider the effect of hierarchy operations on `ua-constraints`.

`AddEdge`$(a, r, r')$ We replace every instance of $r \wedge r'$ in a constraint by $r'$.

`DeleteEdge`$(a, r, r')$ We do nothing since every constraint is a conjunction of elements in an antichain in $R$.

`AddRole`$(a, r, \Delta r, \nabla r)$ We may need to update some constraints because the addition of $r$ will result in the insertion of (transitive) edges between elements in $\Delta r$ and $\nabla r$;

`AddRole`($\texttt{PSO1}, \texttt{X}, \{\texttt{PE1}\}, \{\texttt{QE1}\}$), for example, creates an edge between `PE1` and `QE1`. Therefore, the procedure outlined above for `AddEdge` may need to be employed. (`AddRole`$(a, r, \Delta r, \nabla r)$ may also result in some newly transitive edges being deleted; `AddRole`($\texttt{PSO1}, \texttt{X}, \{\texttt{ENG1}\}, \{\texttt{QE1}\}$), for example, requires the deletion of the transitive edge ($\texttt{ENG1}, \texttt{QE1}$).)

`DeleteRole`$(a, r)$ We replace every constraint of the form $A \cup \{r\}$ in `ua-constraints` by $\overline{A \cup \Delta r}$. An example of this procedure is shown in Figure 4.7.

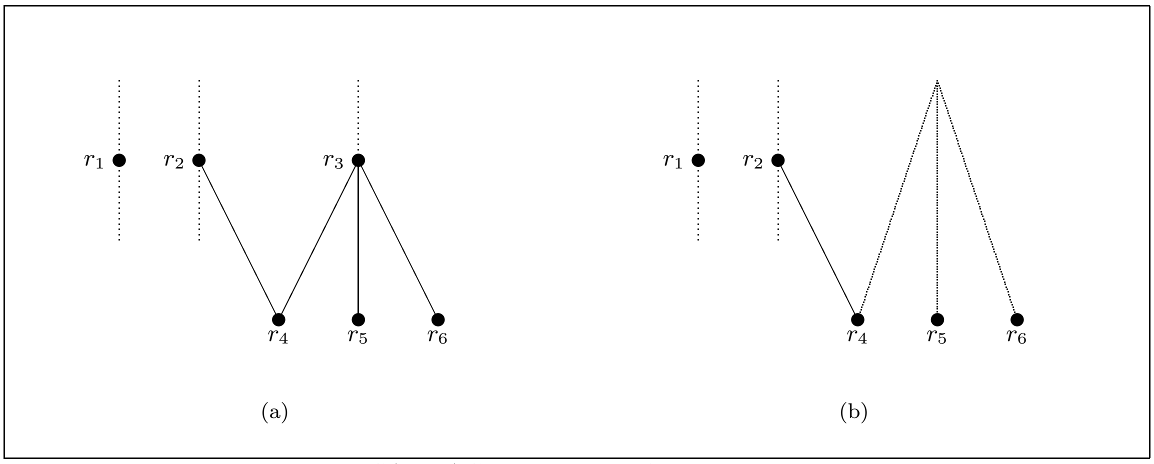


**Figure 4.7:** Role deletion and SARBAC constraints: If $r_3$ is deleted then the constraint $r_1 \wedge r_2 \wedge r_3$ is replaced by $\bigwedge \overline{\{r_1, r_2, r_4, r_5, r_6\}} = r_1 \wedge r_2 \wedge r_5 \wedge r_6$

### Updates to the `pa-constraints` relation

Clearly, similar procedures can be applied to the `pa-constraints` relation: if an edge $(r, r')$ is inserted then every instance of $r \wedge r'$ should be replaced by $r$; if $r$ is deleted from the hierarchy then we replace $A \cup \{r\}$ with $\underline{A \cup \nabla r}$.

### Stronger constraints on user- and permission-role assignment

We briefly consider extending the notion of administrator of a role to an administrator of a user or of a permission. In this case, a role $a$ can add $(u, r)$ to $UA$ if $r \in \sigma(a)$ and $R(u) \subseteq \sigma(a)$. The intuition here is that $a$ is an administrator of $r$ and of the user to which $r$ is to be assigned. Similarly, $a$ can remove $(u, r)$ from $UA$ if $R(u) \subseteq \sigma(a)$. We can define analogous conditions on $p$ and $R(p)$ for permission-role assignment and revocation.

For example, given the user-role assignments and `admin-authority` relation in Figure 4.6, only the `DSO` role would be able to change `anne`'s assignments. It is questionable whether such a scheme is compatible with the requirements of de-centralization and autonomy. For example, assuming the `admin-authority` relation in Table 4.4, `PSO1` would not be able to revoke the assignment of `anne` to `QE1`.

Furthermore, introducing these requirements clearly imposes additional overheads on user- and permission-role assignment and revocation. If we also insist that all users and permissions affected by changes to the role hierarchy made by an administrative role $a$ should be in the administrative scope of $a$, then the overheads increase significantly. In short, although this scheme may be a suitable administrative framework for certain high-assurance applications, it seems unlikely that it would be widely deployed.

### 4.3.2 A comparison of ARBAC97 and SARBAC

In this section we examine the relative merits of ARBAC97 and SARBAC. Table 4.2 summarizes the differences between the two models. The main differences are that only three relations are required in SARBAC rather than the five used in ARBAC97. Furthermore, each of the SARBAC relations is simpler than its ARBAC97 counterpart.

| **ARBAC97** | **SARBAC** |
|---|---|
| `can-assign` | `ua-constraints` |
| `can-revoke` | – |
| `can-assignp` | `pa-constraints` |
| `can-revokep` | – |
| `can-modify` | `admin-authority` |

Table 4.2: A comparison of ARBAC97 and SARBAC features

We now recall some of the observations we made about ARBAC97 in the introductory discussion in Section 4.2 and examine whether SARBAC has any impact on them. Unlike ARBAC97, SARBAC controls the assignment of administrative roles to users. For example, suppose that $(\texttt{PSO1}, \texttt{PL1}) \in$ `ua-constraints`, and we have the `admin-authority` relation and user-role assignments defined in Figure 4.6. Then `claire` acting in the role `DSO` can assign `bill` to `PSO1`.

SARBAC, like ARBAC97 and OASIS, permits a user-role assignment on the basis of the user's current role assignments. However, we believe it is important to impose an *upper bound* or *ceiling* on the roles to which a user can be assigned. (Recall that `dave` could be assigned to `PL1` merely on the basis of his assignment to `ENG1`.) In Chapter 7, we show how such a ceiling can be specified using conflict of interest policies. We consider a different approach to controlling user-role assignment in Chapter 8. (Note we could also consider imposing a "floor" on permission-role assignment in order to prevent sensitive permissions "drifting" too far down the hierarchy.)

SARBAC permits role deletions that may affect SARBAC relations. In fact, SARBAC relations are robust with respect to all role hierarchy operations. SARBAC also permits administration of the SARBAC relations. ARBAC97 makes some limited contribution to considering the effect of hierarchy operations on ARBAC97 relations, but this is neither as comprehensive nor elegant as in SARBAC. In short, we believe SARBAC offers a more complete, self-contained model for administration.

#### Intuitive appeal and simplicity

Obviously the intuitive appeal of a model is a somewhat subjective quality, but we believe that the relationship between SARBAC relations and the role hierarchy is more intuitive in SARBAC than in ARBAC97. In particular, the `admin-authority` relation defines the parts of the hierarchy over which each role has administrative control in a clear and rather natural way. The extended hierarchy makes the division of responsibility for the hierarchy between administrative roles more self-evident than in ARBAC97. SARBAC also benefits from a uniform approach to administrative tasks: they are all performed with reference to the administrative scope of a role. This suggests that an implementation of SARBAC could re-use considerable amounts of code, in contrast to ARBAC97. Furthermore, because SARBAC does not use administrative permissions, no confusion can arise over the purpose of administrative permissions and relations in the administrative model. In short, SARBAC has a more natural, simpler and more streamlined approach to administration than ARBAC97.

Table 4.3 is an informal comparison of the conditions imposed on administrative functions supported by the two models which we described in detail in Sections 3.4, 4.2.4 and 4.3.[1] The purpose of the table is to illustrate the greater coherency of the SARBAC model compared to the ARBAC97 model.

| **Function** | **ARBAC97** | **SARBAC** |
|---|---|---|
| $\texttt{AddRole}(a, r, \Delta r, \nabla r)$ | $\Delta r = \{r'\}$, $\nabla r = \{r''\}$;<br>$(a, R') \in \texttt{can-modify}$;<br>$r', r'' \in R'$;<br>$(r', r'')$ is a create range | $\nabla r \subseteq \sigma(a)$;<br>$\Delta r \subseteq \sigma^+(a)$ |
| $\texttt{DeleteRole}(a, r)$ | $(a, R') \in \texttt{can-modify}$;<br>$r \in R'$;<br>$r$ is not the end point of any range in any ARBAC97 relation | $r \in \sigma^+(a)$ |
| $\texttt{AddEdge}(a, r, r')$ | $(a, R') \in \texttt{can-modify}$;<br>$r, r' \in R'$;<br>$(r) = (r')$ | $r, r' \in \sigma(a)$ |
| $\texttt{DeleteEdge}(a, r, r')$ | $(a, R') \in \texttt{can-modify}$;<br>$r, r' \in R'$ | $r \in \sigma^+(a)$ |
| $\texttt{AssignUser}(a, u, r)$ | $(a, c, R') \in \texttt{can-assign}$;<br>$r \in R'$;<br>$u$ satisfies $c$ | $r \in \sigma(a)$;<br>$(r, c) \in \texttt{ua-constraints}$;<br>$u$ satisfies $c$ |
| $\texttt{RevokeUser}(a, u, r)$ | $(a, R') \in \texttt{can-revoke}$;<br>$r \in R'$ | $r \in \sigma(a)$ |

Table 4.3: A comparison of ARBAC97 and SARBAC functionality

### Computational complexity of SARBAC

An algorithm to test whether $r \in \sigma(a)$ would first construct $\uparrow r$, $\uparrow a$ and $\downarrow a$ ($\mathcal{O}(3|E|)$) and then test whether each element of $\uparrow r$ belongs to $\uparrow a \cup \downarrow a$ ($\mathcal{O}(|R|^2)$). Hence such an algorithm would run in $\mathcal{O}(\max\{3|E|, |R|^2\})$ time. Recall that the time complexity of determining whether a role belongs to a range is $\mathcal{O}(2|E|)$ and of determining whether a given range is encapsulated is $\mathcal{O}(\max\{4|E|, |R|^2\})$.

A detailed comparison of the complexity of implementing ARBAC97 and SARBAC is beyond the scope of this thesis. We have not, for example, considered the complexity of the following:

[1]We have simplified the conditions imposed by ARBAC97 for edge insertion to succeed because of space restrictions.

testing whether a given range is a create range, finding the immediate authority range of a given role and testing whether the insertion of an edge violates the encapsulation of any authority range. However, we can see that the time complexity of the decision problems for ARBAC97 discussed in Section 3.6 will have a similar order in SARBAC. The assignment of users (and permissions) to roles is slower by a multiplicative factor of 1.5 in SARBAC, while hierarchy operations are at least as fast as in ARBAC97. Note that $DP8$ is not relevant in SARBAC because there is no possibility that the `ua-constraints` relation can both permit and prohibit the assignment of a user to a role. We consider $DP7$ and SARBAC in Chapter 5.

**Expressive power**

A rigorous analysis of expressive power is beyond the scope of this thesis. Furthermore, we do not agree with some of the assumptions that informed the development of ARBAC97. In other words, we do not necessarily think it is worthwhile attempting to exhibit a simulation of ARBAC97 using SARBAC techniques. However, we will sketch the differences between ARBAC97 and SARBAC in order to convince the reader that little has been sacrificed to gain the simplicity and versatility of SARBAC. There are two main differences to consider.

Firstly, it is not possible in general to define administration over a range in SARBAC because the administrative scope of a given role will not necessarily be a range. However, it is not obvious to us that the correct unit of administration is a range. In particular, we are unaware of a practical justification for this approach, beyond the requirement that undesirable side effects be avoided. Obviously, a range may be appropriate in certain cases: for example, the range [`ENG1`, `PL1`] is a natural unit of administration for an administrative role with limited powers, but this range is also identified by SARBAC as the correct unit of administration. In short, we believe that SARBAC is a more natural approach to administration and will provide at least as good a solution as ARBAC97 to administration in any given role hierarchy.

Secondly, and perhaps more importantly, it is not possible to define arbitrary constraints on user-role and permission-role assignments. In particular, it is impossible in SARBAC to prohibit the assignment of a user $u$ to a role if $u$ is currently assigned some other specified role. Clearly, such constraints could be used to implement static separation of duty. Moreover, given that such URA97 constraints apply to all users, it would seem that their sole purpose is to implement static separation of duty. It seems to us, therefore, that constraints of the form $\overline{r}$ can be specified using RCL 2000. (Constraints of the form $\overline{r}$ in `can-assignp` can be used to implement operational separation of duty constraints, which can also be specified using RCL 2000.) In short, although constraints of the form $\overline{r}$ provide more options in controlling user-role and permission-role assignments, we would argue that RCL 2000 is a more appropriate way of specifying such behaviour. (It should not be forgotten that constraints of the form $\overline{r}$ give rise to the possibility that the `can-assign` and `can-assignp` relations are not consistent, and also increase the complexity of administration.)

To conclude this section, we note that SARBAC supports administrative functions that are not provided in ARBAC97. In particular, we can update SARBAC relations and constraints and assign administrative roles to users.

## 4.4 Using SARBAC for discretionary access control

Discretionary access control is essentially characterized by ownership of objects by users, and by users having permissions to grant and revoke access to objects they own. It is well known that, in general, the security properties of a discretionary access control system cannot necessarily be established (Harrison et al. 1976).

There have been several attempts to establish a correspondence between features of discretionary access control and role-based access control (Barkley 1997; Friberg and Held 1997; Hua and Osborn 1998; Sandhu and Munawer 1998a). The most ambitious of these attempts, by Osborn et al. (2000), tries to address the problem of ownership in role-based access control. Unfortunately, the construction presented therein involves the creation of at least three administrative roles and one normal role as well as eight permissions for each object in the system. The authors do acknowledge that discretionary access control appears to be more complex to simulate in role-based access control than mandatory access control.

In conclusion, we believe that there is little point in pursuing the simulation of discretionary access control in role-based access control using the methods outlined by Osborn et al. (2000). Indeed, Ferraiolo and Barkley (1997) noted that a discretionary access control system can be viewed as one in which users have administrative (or control) permissions. Taking this point of view, a role-based access control system can be discretionary simply by assigning control permissions to users (via appropriate user-role and permission-role assignments), rather than by assigning distinguished users to administrative roles.

### Auto-administrative roles

We believe a more sensible approach to supporting discretionary access control-style features within a role-based framework would be to have the notion of a "self" role which is "auto-administrative" – that is, a self role can administer itself. We envisage such a role having the following features:

(1) For each user $u$, create a role $r_u$ (possibly with administrative permissions assigned to it) to which only that user is assigned. The only role which can assign users to $r_u$ is $r_u$ itself. (Alternatively, we may wish to make use of cardinality constraints here, and insist that for all $u \in U$, $|U(r_u)| = 1$.)

(2) A user should be able to create roles junior to $r_u$ and to assign users to those roles in order to make permissions available to other users.

(3) Every session a user runs automatically includes that user's self role.

### DRBAC

An obvious solution is to define a set of self roles $R_U$, say, such that $R_U \subseteq \overline{R} \subseteq R$, where $\overline{R}$ is the set of maximal roles in $R$. A role $r_u \in R_U$ could be connected to other roles in the hierarchy and act as a private role (see page 71) so that $u$ also inherits some of the roles to which $u$ is assigned. Alternatively, $r_u$ could be disconnected from the hierarchy and form the root of a "private" hierarchy. For convenience, we refer to these alternatives as DRBAC$_1$ and DRBAC$_2$,

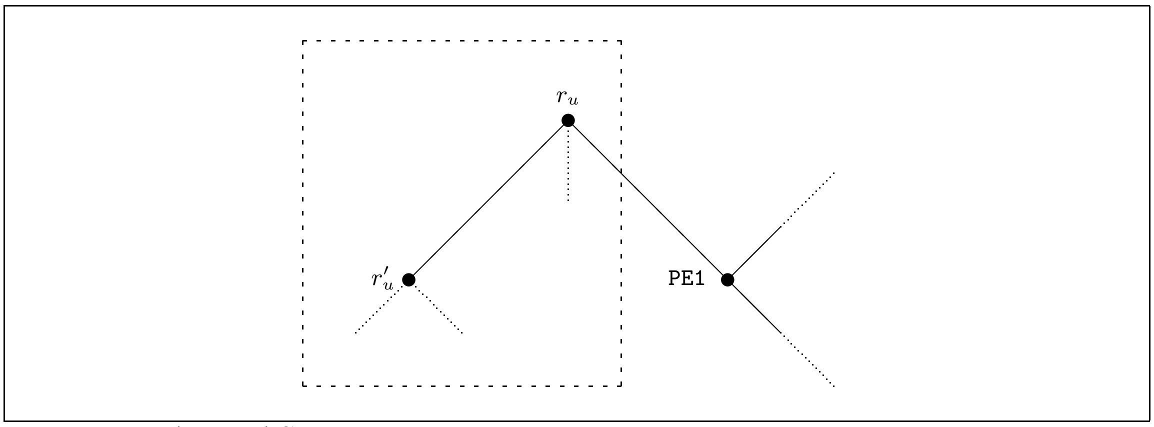


**Figure 4.8:** A DRBAC$_1$ hierarchy: $u$'s private hierarchy is enclosed in the box; note that PE1 is not $r_u$-encapsulated

respectively. Figure 4.8 illustrates DRBAC$_1$ in the context of the usual hierarchy. (Clearly, an illustration of DRBAC$_2$ would be similar but would not include the edge $(\texttt{PE1}, r_u)$.)

RHA$_4$ provides a natural way of administering self roles and realizing features (1) and (2). Specifically, for every $r_u \in R_U$, $(r_u, r_u) \in$ `admin-authority`. Hence $u$ will be able to create role $r'_u$ with $r_u$ as a parent and assign permissions and users to $r'_u$. The administrative scope of $r_u$ will be a private hierarchy within which $u$ can exercise discretionary powers. (Clearly (3) is merely an implementation decision.)

There are two issues that may require attention if we choose to implement DRBAC$_1$. Firstly, if a user $u$ changes job and hence requires different role assignments, it may be necessary to change the way in which $r_u$ is connected to the hierarchy. This is not possible unless there exists $a \in R$ such that $r_u \in \sigma(a)$. Since $r_u$ is a maximal element, this implies that $(a, r_u) \in$ `admin-authority` (although this does not imply that $a$ inherits the permissions of $r_u$). Secondly, because $(r_u, r_u) \in$ `admin-authority`, there may be $r \in R$ such that $r \in \sigma(r_u)$. Hence $r_u$ may be able to administer parts of the main role hierarchy. In practice, this seems unlikely to happen (see PE1 in Figure 4.8, for example, which does not belong to $\sigma(r_u)$).

Note that $r_u$ in Figure 4.8 is identical to the concept of a private role as suggested by Sandhu et al. (1996), which was introduced to address the overloading of the hierarchy. However, RRA97 is inapplicable to the role hierarchy depicted in Figure 4.8 because of the requirement for encapsulated ranges. Furthermore, RRA97 does not permit the creation of a role that has no immediate child, like $r'_u$, for example.

## Concluding remarks

We suggested earlier that the self role could always be activated at login. If such an approach is taken, DRBAC$_1$ suggests an alternative approach to user-role assignment in role-based access control. We could use a dynamic approach to user-role assignment that captures the principle of least privilege by dispensing with the *UA* relation and forcing a user to activate roles. A request to activate a role will be granted if an appropriate constraint in the `ua-constraints` relation is satisfied. This approach to user-role assignment is clearly very similar to that adopted in OASIS. However, unlike in OASIS, we retain the possibility of using the role hierarchy to determine permission inheritance. The details of such an approach are beyond the scope of this thesis.

## 4.5 The generalized role-based access control model

Sandhu and Munawer (1998b) noted in their discussion of RRA97 that a role $r$ could be regarded as an *ability* if no users were assigned to $r$, and a role $r'$ could be regarded as a *group* if no permissions were assigned to $r'$. In fact, RRA97 consists of three sub-models, GRA97, ARA97 and UP-RRA97, which model group-role assignment, ability-role assignment and role-role assignment, respectively. GRA97 and ARA97 are essentially generalizations of URA97 and PRA97, respectively. What we described as RRA97 is actually UP-RRA97, which models the administration of roles which have both users and permissions assigned to them.

An important topic for research is the ability hierarchy. Although it seems attractive to order the set of abilities, it is not immediately obvious what this hierarchy should be. The most obvious approach is to treat abilities in the same way as roles, which is the approach advocated in Sandhu and Munawer (1998b). An example of such an ability hierarchy is shown in Figure 4.9a. A primitive permission is denoted by a subscripted $p$; an ability is denoted by a subscripted $\pi$.

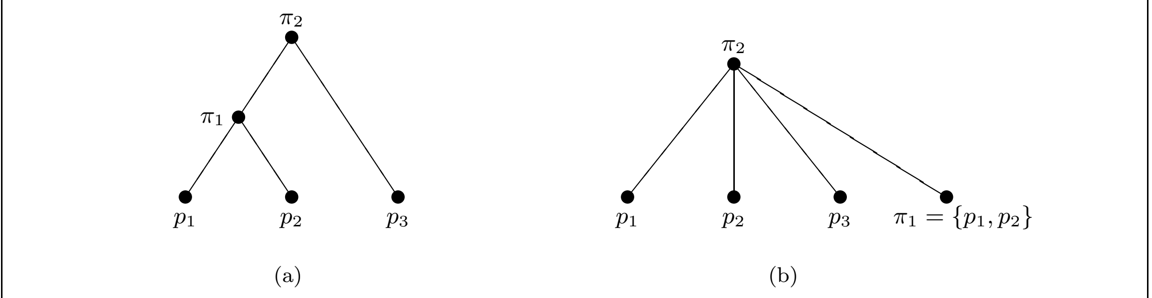


**Figure 4.9:** Ability hierarchies

Consider the following question: if a role $r$ is assigned the ability $\pi_1 = \{p_1, p_2\}$, should a request by (a user assigned to) $r$ to use permission $p_1$ be granted? Clearly the answer to this question is "yes" given the underlying assumption of the hierarchy in Figure 4.9a. However, it is less obvious that the answer should be "yes". There may well be applications in which an ability should be a "black box", in order to implement a particular feature of the application, and that individual permissions in that ability should not be available. For example, suppose that the permissions $p_1$ and $p_2$ should not be assigned implicitly as atomic permissions to $\pi_1$. This results in a different hierarchy, shown in Figure 4.9b. (Note that we have assigned $\pi_1$ to $\pi_2$ as in Figure 4.9a.) Unfortunately, it now becomes more difficult to define a natural partial ordering on the set of abilities.

Another important issue is the interaction between the ability hierarchy and the ability-assignment relation. In common with RBAC96, most role-based access control models assume that complex permissions can be formed from other permissions.

However, suppose we have the ability hierarchy shown in Figure 4.9a, and that $PA = \{(\pi_1, \texttt{PE1}), (\pi_2, \texttt{ENG1})\}$. Clearly $PA$ contains redundancy: the permissions explicitly assigned to `PE1` are already implicitly assigned through $(\pi_2, \texttt{ENG1})$. In a large system, redundancy in an access control system imposes a significant overhead and should be avoided where possible (O'Shea 1995).

Therefore, it may be desirable to impose an additional constraint on ability-role assignment. Specifically, for all $p, p' \in P$, if $p \leqslant p'$, then for all $r \in R(p)$, $r' \in R(p')$, $r \not> r'$. In other words, no element in the set of roles to which $p$ is assigned can be senior to a role in the set of roles to which $p'$ is assigned.

### 4.5.1 The generalized role hierarchy

We believe that the incorporation of groups and abilities into role-based access control models could be explored more fully. We now present a preliminary discussion of how these concepts may be brought together in a "generalized role-based access control model".

We define a group to be a non-empty set of users. The set of groups, $G$, is a partial order under superset inclusion, which we denote $\langle G, \leqslant_{\mathtt{g}} \rangle$. (A user is modelled as a group with a single element.)

Permissions are considered to be "uninterpreted symbols" as usual, although we assume the existence of *primitive* or *atomic* permissions. We define an ability to be a non-empty set of permissions. We will assume the natural interpretation of the ability hierarchy, as shown in Figure 4.9a, for example. That is, the set of abilities, $P$, is a partial order under subset inclusion, which we denote $\langle P, \leqslant_{\mathtt{p}} \rangle$.

We now describe a generalized role hierarchy which we believe to be a simple, intuitive generalization of existing approaches to the role hierarchy. Given a role hierarchy, $\langle R, \leqslant_{\mathtt{r}} \rangle$, a group-role assignment relation, $UA$, and an ability-role assignment relation, $PA$, we construct a partially ordered set, $\langle H, \leqslant \rangle$, where $H = G \cup R \cup P$.

In order to simplify our presentation we introduce the notion of type before we define $\leqslant$. We denote the *type* of $h \in H$ by $\tau(h)$, where

$$\tau(h) = \begin{cases} \mathtt{g} & \text{if } h \in G, \\ \mathtt{r} & \text{if } h \in R, \\ \mathtt{p} & \text{if } h \in P. \end{cases}$$

In addition, the set of types is a total order, where $\mathtt{p} < \mathtt{r} < \mathtt{g}$.

We define $\leqslant$ to be the reflexive, transitive closure of the following covering relation. For $h, k \in H$, $h \lessdot k$ if one of the following conditions holds:

$$\tau(h) = \tau(k) \text{ and } h <_{\tau(h)} k;$$
$$\tau(h) = \mathtt{p},\ \tau(k) = \mathtt{r} \text{ and } (h, k) \in PA;$$
$$\tau(h) = \mathtt{r},\ \tau(k) = \mathtt{g} \text{ and } (k, h) \in UA.$$

The generalized role hierarchy is the graph $\langle H, \lessdot \rangle$. For all $h \in H$, there exists a set of abilities, $\downarrow P(h)$, which denotes the abilities (implicitly) assigned to $h$. Furthermore, for all $h, k \in H$, we have $h < k$ implies either $\downarrow P(h) \subset \downarrow P(k)$ or $\downarrow P(h) \subseteq \downarrow P(k)$ and $\tau(h) < \tau(k)$.

We now consider an example based on fragments of the role hierarchy and assignment relations of Section 3.1. The resulting generalized role hierarchy is shown in Figure 4.10. All abilities in this example are primitive permissions. The element `QEs` represents the group of (users that are) quality engineers.

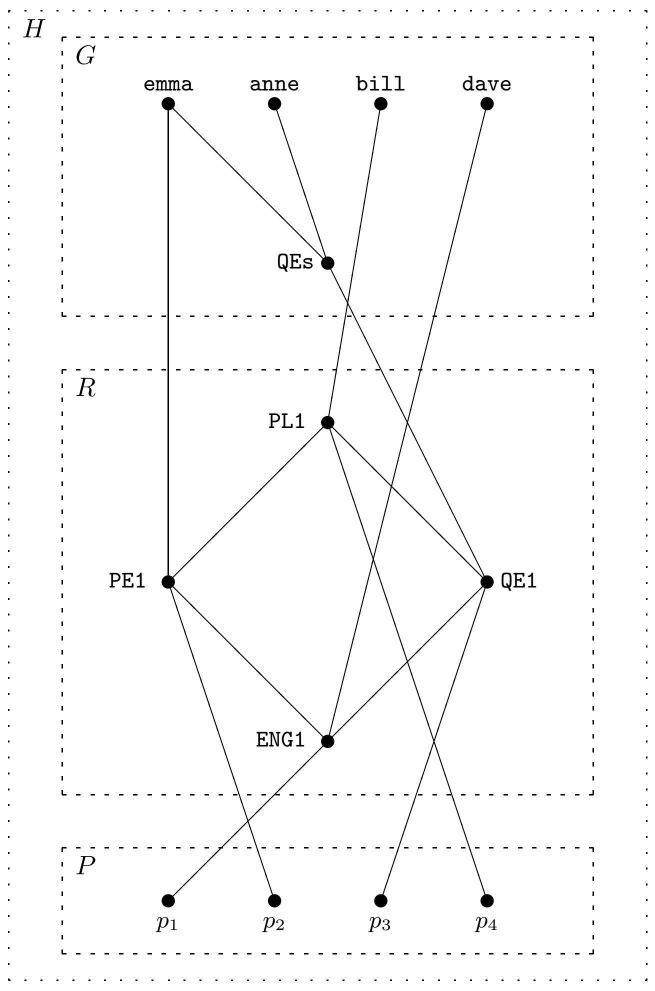


**Figure 4.10:** A generalized role hierarchy

We note that per-subject, per-permission and per-ability reviews effectively become reachability problems in a directed graph. For example, the set of permissions assigned to a group $g$ (recall that the set of groups includes all single users), is simply the union of the abilities in $\downarrow g$. Similarly, to establish the set of groups which can exercise a given permission $p$, we compute the set of groups in $\uparrow p$. (In other words, the generalized role hierarchy model conforms to the requirements of symmetrical RBAC – the top level in the NIST RBAC model.) Another attractive feature of this hierarchy is that no confusion can arise in the assignment of permissions to roles. Recalling the ability hierarchy of Figure 4.9a, assuming the access control system stores the transitive reduction of the generalized hierarchy, then it will contain edge $(\pi_2, \texttt{ENG1})$ but not $(\pi_1, \texttt{PE1})$.

We note that the generalized role-based access control model bears some similarity to the directed graph model of protection systems (Snyder 1981). In this model, there are two types of vertices representing subjects and objects, depicted by solid and hollow circles, respectively. An edge is drawn between two vertices $v_1$ and $v_2$ and labelled $\alpha$ if subject $v_1$ has access rights

$\alpha$ to object $v_2$. Existing results (Lipton and Snyder 1977) may simplify the task of establishing properties of the generalized role-based access control model.

### 4.5.2 Administration and the generalized role hierarchy

One beneficial side-effect of the generalized role hierarchy is that user- and permission-role assignment are hierarchy operations. In particular, user- and permission-role assignment corresponds to inserting an edge in the generalized role hierarchy; user- and permission-role revocation corresponds to deleting an edge from the generalized role hierarchy. User- and permission-role assignment are controlled by the RHA$_4$ model and the SARBAC relations `ua-constraints` and `pa-constraints`. For example, given $p \in P$ and $r \in R$, by (4.3), $p$ can be assigned to $r$ by $a$ provided $p, r \in \sigma(a)$ (and $p$ satisfies an appropriate SARBAC constraint).

Two interesting points arise when applying SARBAC to the generalized role hierarchy. Firstly, we can have tuples of the form $(u, u')$ in the `admin-authority` relation, where $u, u' \in U$. Hence $u$ can manage the assignments of roles and permissions to $u'$. Secondly, we can have constraints in the `ua-constraints` relation of the form $G_1 \wedge \cdots \wedge G_k$, where $G_i$, $1 \leqslant i \leqslant k$, is a group. A more detailed examination of these topics is beyond the scope of the thesis.

## 4.6 Summary and discussion

We believe that the ARBAC97 model has certain shortcomings: it suffers from a lack of applicability, flexibility, coherence and robustness; the interaction between its sub-models is not completely determined; it is rather complex and lacks intuitive appeal. We believe that many of these problems arise because of two particular features in the development of the ARBAC97 model.

Firstly, we believe that a sensible approach to the problem of administration in role-based access control is to first determine how hierarchy operations are to be performed. However, in ARBAC97, the "easy" models, URA97 and PRA97, were developed first; as a result of this, the integration of these models with RRA97 has not been easy to achieve.

Secondly, the development of RRA97 was based on encapsulated ranges. The reason for this decision was that the model should support decentralization, autonomy and should not allow undesirable changes to the hierarchy. It is clear that the decision to develop an administrative model that requires the existence and preservation of encapsulated ranges in a role hierarchy has had a detrimental effect on the applicability of the resulting model. In other words, the development of ARBAC97 has been driven by the concept of an encapsulated range not by the needs of RBAC96; surely this is the wrong way round. Furthermore, although RRA97 guarantees that changes are local and cannot propagate undesirable changes through the hierarchy, it is not obvious that the changes which are deemed to be undesirable have any formal or intuitive basis.

We believe that the list of requirements for ARBAC97 was missing several important items. For example, surely ARBAC97 should be applicable to as many role hierarchies as possible. We believe it is more appropriate to develop a more permissive model that does not preclude administrative changes because they conflict with the assumptions of the administrative model. Rather, the model should permit changes if they are reasonable in some intuitive sense. (This requirement is no more vague than the requirement that RRA97 should not permit undesirable changes.) Clearly, such a model should provide support for accountability and should be easily integrated with RBAC96.

The RHA family of models is a set of four increasingly complex administrative models for hierarchy operations. Each of these models is based on the idea of administrative scope, which bears some formal similarity to an encapsulated range.

We have concentrated on applications of the RHA$_4$ model as it provides a realistic compromise between utility and complexity. It is also the model that can be most easily compared with RRA97. We have shown that RHA$_4$ addresses the shortcomings of RRA97, although it is considerably more permissive that RRA97. (However, the concept of a line manager role provides direct support for accountability, a feature that is not made explicit in RRA97. Unfortunately, the introduction of the `admin-authority` relation to RHA$_4$ means that the definition of line manager needs to be modified for RHA$_4$. We discuss this issue further in Chapter 9.) The fact that RHA$_4$ is applicable to many different types of role hierarchy means that it can also be used to reduce the amount of inheritance in the role hierarchy, thereby addressing certain reservations (reviewed in Section 4.1.1) that have been expressed about the suitability of the role hierarchy for modelling access control requirements in an enterprise.

The benefit of defining the RHA family of models is that each of them can be extended to a complete model for administration. In this chapter we extended RHA$_4$ to the SARBAC model by introducing the `ua-constraints` and `pa-constraints` relations which control the assignment of roles to users and permissions, respectively. These relations are analogous to `can-assign` and `can-assignp`, and make use of SARBAC constraints. These constraints are defined in terms of antichains and are satisfied according to the same criteria as URA97 and PRA97 constraints. However, SARBAC constraints are considerably simpler, allowing neither disjunctions nor constraints of the form $\overline{r}$, thereby making the management of the SARBAC relations easier. In particular, a simple algorithm can update the SARBAC relations in the event of changes to the role hierarchy, a feature which has not been considered in ARBAC97.

A useful side effect of SARBAC is the support it provides for private user hierarchies, thereby providing support for discretionary access control features within a role-based environment. The resulting DRBAC (discretionary RBAC) model is far simpler than any previous attempts to simulate or support discretionary access control in a role-based access control model.

We have also presented a generalized role-based access control model in which all entities are treated as nodes in a hierarchy. The set of entities may include groups (a set of users) and abilities (a set of permissions). Edges in the hierarchy indicate assignment of a group to a role, or of a role to a role, or of a role to an ability. The SARBAC model can be employed with the generalized role-based access control model.

There are many opportunities for further work in the area of administration in role-based access control, which we will discuss in Chapter 9. Probably the most interesting practical development would be to implement a role-based reference monitor that could use one of several different administrative models (including ARBAC97 and SARBAC) and to assess the relative performance of these models.

# Chapter 5

# The Safety Problem

Informally, the safety problem is the determination of whether or not a given subject can acquire a particular access mode to a given object. In a practical sense, the safety problem is "given a specification of security (an access control policy) does the implementation coincide with policy?". Informally, given an access control model, the harder the safety problem is for that model, the weaker the safety properties of the model. For example, the safety problem in the protection matrix model is undecidable in general. Therefore, we say that the protection matrix model has weak security properties, and that the Bell-LaPadula model, for example, has strong security properties.

The main contribution of this chapter is to prove that the safety problem in the RBAC96/ARBAC97 model is undecidable. In the next section we review the safety problem in the HRU model in order to introduce some of the techniques that have been brought to bear on the topic. In Section 5.2 we consider the safety problem in role-based access control. We prove that the safety problem in RBAC96 and in RBAC96/ARBAC97 is undecidable.[1] Finally, in Section 5.2.3 we consider the safety problem for RBAC96/SARBAC. Unfortunately, it is still undecidable, although the analysis is considerably simpler than that for RBAC96/ARBAC97.

## 5.1 The protection matrix model

We first state the safety problem and seminal results for the protection matrix model (Harrison et al. 1976; Harrison and Ruzzo 1978). Given a protection system $\mathsf{S}$, the question of whether an access right $a \in A$ can be entered into a cell of the matrix (which did not previously contain it) through some sequence of commands in $\Gamma$ is called the *safety problem*. Formally, does there exists a sequence of commands, $\gamma_1, \dots, \gamma_n$, where $\gamma_i \in \Gamma$, $1 \leqslant i \leqslant n$, giving rise to a sequence of matrices $M_0, \dots, M_n$, where $M_i : S_i \times O_i \to 2^A$, such that there is a command $\gamma$ whose conditional part is satisfied by $M_n$ and whose body contains the operation `enter` $a$ `in [`$s$`,`$o$`]` for some $s \in S_n, o \in O_n$. The action of entering $a$ into a cell of the matrix is called a *leakage*. Although "leakage" seems to be a somewhat pejorative term, it should be noted that any access

[1]It is relatively easy to prove that the safety problem for RBAC96 is undecidable because RBAC96 can be simulated using a variant of the protection matrix model that is known to have an undecidable safety problem (Munawer and Sandhu 1999).

control mechanism of interest must permit certain leakages otherwise no changes can be made to the authorizations in the system. We now state the seminal results on the safety problem for the protection matrix model, which we will denote **SP:HRU**.

**Theorem 5.1.1 (Harrison et al. 1976, Theorem 2)** ***SP:HRU*** *is undecidable.*

**Proof** (Sketch) The proof proceeds by showing that the behaviour of an arbitrary DTM with alphabet $\Sigma$ and set of states $Q$ can be simulated by a protection system $\mathsf{S}(\Sigma \cup Q, \Gamma, M_0)$, and that the leakage of a right corresponds to the DTM entering the halt state, $q_h$. By construction, the protection matrix $M : S \times S \to 2^{\Sigma \cup Q}$ is square. For a DTM in configuration $a_1 a_2 \dots a_{k-1} q a_k \dots a_m$, the entries in the matrix are defined as follows:

$$[s_i, s_j] = \begin{cases} \{a_i, q\} & \text{if } i = j = k \neq m, \\ \{a_i\} & \text{if } i = j \neq k, \\ \{a_i, \texttt{end}\} & \text{if } i = j = m, \\ \{\texttt{own}\} & \text{if } j = i + 1, \\ \emptyset & \text{otherwise.} \end{cases}$$

The matrix in Figure 5.1a simulates a DTM in configuration $a_1 q a_2 a_3 a_4$. The generic right `own` is used to order the subjects so that the $i$th subject is identified with the contents of the $i$th square of the tape. The generic right `end` is used to identify the end of the tape.

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ |
|---|---|---|---|---|
| $s_1$ | $\{a_1\}$ | $\{\texttt{own}\}$ | | |
| $s_2$ | | $\{a_2, q\}$ | $\{\texttt{own}\}$ | |
| $s_3$ | | | $\{a_3\}$ | $\{\texttt{own}\}$ |
| $s_4$ | | | | $\{a_4, \texttt{end}\}$ |

(a) Configuration $a_1 q a_2 a_3 a_4$

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ |
|---|---|---|---|---|
| $s_1$ | $\{a_1\}$ | $\{\texttt{own}\}$ | | |
| $s_2$ | | $\{a'_2\}$ | $\{\texttt{own}\}$ | |
| $s_3$ | | | $\{a_3, q'\}$ | $\{\texttt{own}\}$ |
| $s_4$ | | | | $\{a_4, \texttt{end}\}$ |

(b) Configuration $a_1 a'_2 q' a_3 a_4$

**Figure 5.1:** Simulating a Turing machine using a protection matrix

For each state transition of the DTM, $\delta(q, a)$, we define a command in $\mathsf{S}$, $c_{aq}$, that simulates the behaviour of the state transition. Command 5.1 simulates the state transition $\delta(q, a_2) = (q', a'_2, 1)$. The comments show the actual parameters that satisfy the conditional part of the command and how the command identifies the state transition $\delta(q, a_2)$.

By construction, for a given configuration of the protection matrix, there is only one command whose conditional expression is satisfied, and one combination of subjects that satisfies the conditional part of that command (which will be a pair of the form $(s_i, s_{i+1})$, for some $i$, $1 \leqslant i < |S|$). In Figure 5.1a that command is $c_{a_2 q}$ and the pair of subjects is $(s_2, s_3)$. The resulting protection matrix is shown in Figure 5.1b. Clearly, if the DTM enters $q_h$, then $q_h$ is entered into the protection matrix. ■

**Command 5.1**

```
$c_{a_2q}(s, s')$                                                    /* $(s_2, s_3)$ */
    if
        $q \in [s, s]$ and                                        /* identifies $s_2$ */
        own $\in [s, s']$ and                                     /* identifies $s_3$ */
        $a_2 \in [s, s]$                     /* identifies state transition $\delta(q, a)$ */
    then
        delete $q$ from $[s, s]$
        delete $a_2$ from $[s, s]$
        enter $q'$ in $[s', s']$
        enter $a_2'$ in $[s, s]$
```

**Theorem 5.1.2 (Harrison et al. 1976, Theorem 1)** ***SP:HRU*** *for mono-operational protection systems is* ***NP****-complete.*

**Proof** (Sketch) The proof establishes that it is sufficient to consider enter commands in a mono-operational protection system, and that the maximum length of a sequence of commands leading to a leakage is proportional to the size of the initial matrix. The decision procedure merely examines all possible sequences of commands in order to decide the safety problem. ■

Unfortunately, a mono-operational protection system is of very limited practical use since it is impossible to both create a subject (or object) and to enter any access rights in that subject's row of the matrix (or that object's column). Since conditional commands test for the presence of access rights, no newly-created subject can ever satisfy the conditions of a conditional command, and is therefore effectively powerless.

**Theorem 5.1.3 (Harrison and Ruzzo 1978, Theorems 5–8)** ***SP:HRU*** *for monotonic protection systems is decidable if the protection system is mono-conditional and undecidable otherwise.*

**Proof** (Sketch) The proof proceeds by demonstrating that the safety problem for monotonic protection systems is equivalent to the Post correspondence problem, which is undecidable in general (Post 1946). It can be shown there is a set of commands, several of which use more than one conditional statement, which encode the Post correspondence problem in a protection matrix. The leakage of a generic access right is equivalent to finding a solution to the Post correspondence problem.

The proof that a mono-conditional protection system is decidable proceeds by showing that there is a *chain* of minimal length which leaks a generic right. A chain is a sequence of commands, $\gamma_1, \ldots, \gamma_k$, in which the conditional statement in $\gamma_i$ is satisfied by an access right which is inserted into the matrix by $\gamma_{i-1}$, $1 \leqslant i \leqslant k$. Moreover, the length of the chain is $\mathcal{O}(|A|)$. The proof also introduces the notion of an ordering on protection matrices. This ordering is referred to as a *covering* in a general protection system, and as a *weak covering* in a monotonic protection system. ■

It is an open problem whether the safety problem in (non-monotonic) mono-conditional protection systems is decidable in general. To conclude this section, we summarize the known results for **SP:HRU** in Table 5.1.

| **Operations** | **Mono-conditional** | **Mono-operational** | **Status** | **Construction** | **Reference** |
|---|---|---|---|---|---|
| $\{op_1, op_2, op_3, op_4, op_5, op_6\}$ | ✗ | ✗ | undecidable | Halting problem | Harrison et al. 1976, Theorem 2 |
| $\{op_1, op_2, op_3, op_4, op_5, op_6\}$ | ✗ | ✓ | **NP**-hard | Length of shortest "leaky" computation is bounded by size of $M_0$ | Harrison et al. 1976, Theorem 1 |
| $\{op_1, op_4, op_5, op_6\}$ | ✗ | ✗ | **PSPACE**-complete | Any polynomial space bounded Turing machine is equivalent to some set of commands and $M_0$ whose size is polynomial in the size of the Turing machine input | Harrison et al. 1976, Theorem 3 |
| $\{op_1, op_3, op_4, op_6\}$ | ✗ | ✗ | decidable | Covering problem in vector addition systems | Lipton and Snyder 1978 |
| $\{op_1, op_2, op_3\}$ | ✗ | ✗ | undecidable | Post correspondence problem | Harrison and Ruzzo 1978, Theorem 5 |
| $\{op_1, op_2, op_3\}$ | ✓ | ✗ | **NP**-hard | Length of shortest "leaky" computation is $\mathcal{O}(\lvert A \rvert)$ | Harrison and Ruzzo 1978, Theorem 8 |
| $\{op_1, op_2, op_3, op_4\}$ | ✓ | ✗ | decidable | Generalization of result for $\{op_1, op_2, op_3\}$ | Harrison and Ruzzo 1978, Theorem 9 |

Table 5.1: The safety problem in the protection matrix model: To simplify the presentation, we denote the primitive operations `enter`, `create subject`, `create object`, `delete`, `destroy subject` and `destroy object` by $op_1, \ldots, op_6$, respectively. For example, a protection system whose commands only use the operations $op_1$, $op_2$ and $op_3$ is monotonic.

## 5.2 Role-based access control

We will write **SP:RBAC** to denote the safety problem in role-based access control. **SP:HRU** decides whether it is possible for an access right to be added to an element of the protection matrix which does not currently contain that access right. We adopt the following definition of the safety problem in the context of role-based access control. **SP:RBAC** decides whether it is possible for a permission to be assigned to a role which does not currently have that permission assigned to it. There are two related problems:

- Is it possible for a user $u$ to be assigned a role that $u$ is not currently assigned to?
- For a given permission $p$ and a given user $u$, is it possible that some sequence of permission-role assignments and user-role assignments results in $(u, r) \in UA$ and $(p, r) \in PA$ for some $r \in R$? (In other words, at some point in the future, is it possible that $u$ can exercise permission $p$?)

We first prove that the safety problem using the RBAC96 model is undecidable, and use this result to prove that the safety problem using the RBAC96/ARBAC97 model is also undecidable. To conclude this section we prove that the safety problem using the RBAC96/SARBAC model is also undecidable.

We base our analysis on RBAC96 and ARBAC97 because they form the most sophisticated and comprehensive role-based access control models. Therefore, we do not have to make many assumptions about how the model should work. (Harrison et al. (1976) took several pages to establish the framework within which their analysis of the safety problem would take place.) Nevertheless we will have to give some operational semantics to the RBAC96/ARBAC97 model. Specifically we will need to introduce the idea of *commands* which can change the components of the models.

### 5.2.1 RBAC96

A *role-based protection system* $\mathsf{S}(P, \Gamma, R_0, UA_0, PA_0, RH_0)$ is defined by a set of permissions $P$, a set of commands $\Gamma$, an initial role hierarchy $R_0$ and initial RBAC96 relations $UA_0$, $PA_0$ and $RH_0$. A command consists of a conditional statement comprising a conjunction of zero or more logical expressions, and a body consisting of a sequence of one or more primitive operations. We postulate the following primitive operations:

- `add-user-assignment` adds a single tuple to the $UA$ relation;
- `remove-user-assignment` removes a single tuple from the $UA$ relation;
- `add-permission-assignment` adds a single tuple to the $PA$ relation;
- `remove-permission-assignment` removes a single tuple from the $PA$ relation;
- `add-role` adds a role to $R$;
- `add-edge` adds an edge to $RH$.

The body of a command is executed only if the conditional statement evaluates to true. A logical expression may test whether a particular permission-role assignment is in $PA$ (analogous to testing the presence of an access right in a cell of the protection matrix). However, it may be a more complicated test, checking whether two roles are the end points of a create range, for example.

**Theorem 5.2.1** *The safety problem for RBAC96 is undecidable.*

**Proof** We now outline the transformation of the halting problem to **SP:RBAC**, demonstrating the construction through examples of role hierarchies and commands. We split the proof into sections describing how each of the basic concepts in RBAC96 is used in the transformation.

**Permissions and the permission-role assignment relation** We identify tape symbols and states of the DTM with permissions and use permission-role assignments to keep track of the read-write head.

We use the permission $e$ to mark the rightmost non-blank cell of the tape. The permission-role assignments $(\overline{r}_i, a_i)$, $i \in \mathbb{N}^+$, allow the administrative role $a_i$ to assign and revoke permissions to $r_i$. The permission-role assignments $(\overline{\overline{r}}_{i+1}, a_i)$, $i \in \mathbb{N}^+$, allow the administrative role $a_i$ to assign permissions to $r_{i+1}$. (The permission-role assignments $(\overline{r}_i, a_i)$ and $(\overline{\overline{r}}_{i+1}, a_i)$ perform a similar function to the generic right `own` $\in [s_i, s_{i+1}]$ in the proof of **SP:HRU**.) We also assume that $a_\top$ has the permissions to change the permission assignments of subordinate administrative roles.

For example, suppose that the initial configuration of the DTM is $q_1p_1p_2p_3p_4$. Then we define

$$PA = \{(q_1, r_1), (p_1, r_1), (p_2, r_2), (p_3, r_3), (p_4, r_4), (e, r_4)\} \cup APA, \text{ where}$$
$$APA = \left\{(\overline{r}_1, a_1), (\overline{\overline{r}}_2, a_1), (\overline{r}_2, a_2), (\overline{\overline{r}}_3, a_2), (\overline{r}_3, a_3), (\overline{\overline{r}}_4, a_3), (\overline{r}_4, a_4)\right\} \cup \left\{(\overline{a}_i, a_\top) : i \in \mathbb{N}^+\right\}.$$

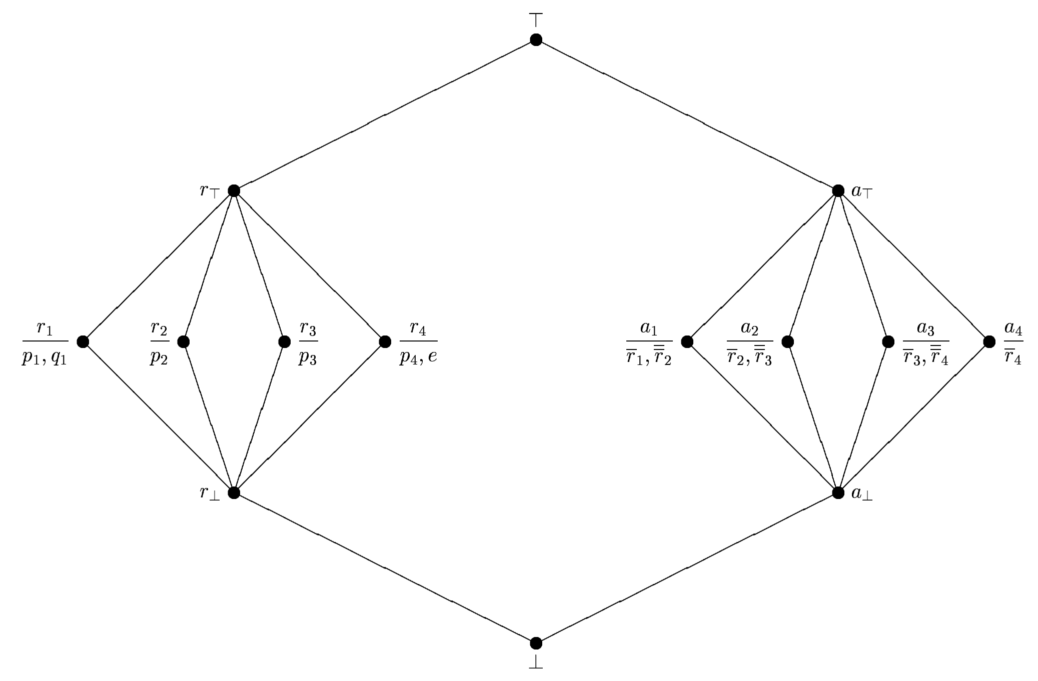
**Roles and the role hierarchy** The role hierarchy is used to organize the roles and administrative roles in a way that corresponds to the contents of the Turing machine tape. Figure 5.2a shows the role hierarchy corresponding to the configuration $q_1p_1p_2p_3p_4$.[2]

In order to clarify our construction we have labelled the nodes with both the role name and the permissions assigned to that role. Note that the ranges $[r_\perp, r_\top]$ and $[a_\perp, a_\top]$ are isomorphic under the mapping $r_x \mapsto a_x$, $x \in \mathbb{N}^*$, where $\mathbb{N}^* = \{\perp, \top\} \cup \mathbb{N}^+$. Therefore, we may refer informally to a role *corresponding* to an administrative role.

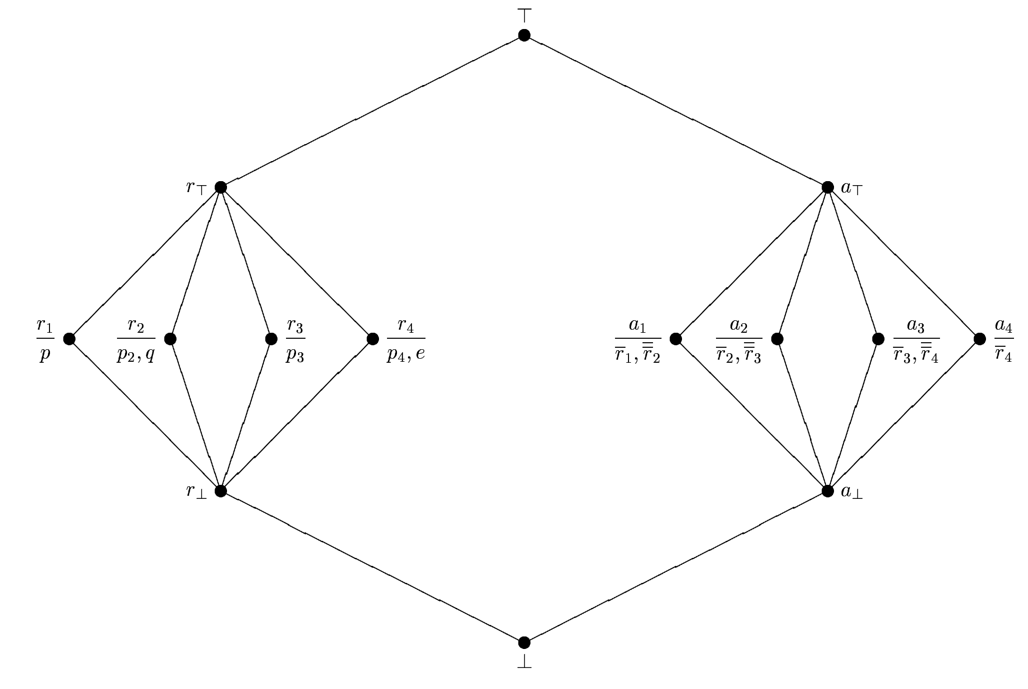
**Commands and operations** Finally, we illustrate how the state transition function $\delta$ is simulated using commands. For each possible state transition $\delta(p, q)$, a command $C_{pq}$ is defined satisfying the following properties:

- its conditional statement is satisfied by a unique tuple of parameters;
- its body consists of operations which preserve the correspondence between the DTM and the role hierarchy.

[2]Note that this hierarchy is more complicated than is actually required in order to prove the result in the context of RBAC96. However, the RRA97 component of ARBAC97 imposes restrictions on the structure of the role hierarchy. Therefore, we have chosen to use the same role hierarchy for the proof of the undecidability of **SP:RBAC** in the context of both RBAC96 and RBAC96/ARBAC97.


$\top$
$r_\top$
$a_\top$
$\frac{r_1}{p_1, q_1}$
$\frac{r_2}{p_2}$
$\frac{r_3}{p_3}$
$\frac{r_4}{p_4, e}$
$\frac{a_1}{\overline{r}_1, \overline{\overline{r}}_2}$
$\frac{a_2}{\overline{r}_2, \overline{\overline{r}}_3}$
$\frac{a_3}{\overline{r}_3, \overline{\overline{r}}_4}$
$\frac{a_4}{\overline{r}_4}$
$r_\perp$
$a_\perp$
$\perp$

(a) Role hierarchy for configuration $q_1p_1p_2p_3p_4$


$\top$
$r_\top$
$a_\top$
$\frac{r_1}{p}$
$\frac{r_2}{p_2, q}$
$\frac{r_3}{p_3}$
$\frac{r_4}{p_4, e}$
$\frac{a_1}{\overline{r}_1, \overline{\overline{r}}_2}$
$\frac{a_2}{\overline{r}_2, \overline{\overline{r}}_3}$
$\frac{a_3}{\overline{r}_3, \overline{\overline{r}}_4}$
$\frac{a_4}{\overline{r}_4}$
$r_\perp$
$a_\perp$
$\perp$

(b) Role hierarchy for configuration $pqp_2p_3p_4$

**Figure 5.2:** Role hierarchies in RBAC96/ARBAC97 for **SP:RBAC**

Therefore, since $\delta$ is a function, the above conditions ensure that the role hierarchy and permission-role assignment relation will mimic the behaviour of the DTM.

We consider the case $\delta(q,p) = (q',p',1)$ in which a state transition causes the read-write head to move right. There are two cases to consider.

- The new cell does not contain $b$ (the blank symbol)

  We define the generic command $C_{pq}$ in Command 5.2. The comments show the actual parameters that satisfy the conditional part of the command $C_{p_1q_1}$, given $\delta(p_1,q_1) = (q,p,1)$. With reference to Figure 5.2a, and assuming there is a state transition $\delta(q_1,p_1) = (q,p,1)$, the logical tests of the corresponding command $C_{p_1q_1}$ are only satisfied by the parameters in the ordered triple $(a_1,r_1,r_2)$. Lines 03 and 04 fix the unique role which satisfies the conditional part of the command and the unique state transition that the command simulates. Lines 05 and 06 fix the unique administrative role which satisfies the conditional part of the command and also establish the role to which the new state will be assigned. The resulting hierarchy is shown in Figure 5.2b.

**Command 5.2**

```
01    $C_{pq}$($a$ role, $r$ role, $r'$ role)                          /* ($a_1$,$r_1$,$r_2$) */
02        if
03            ($q$,$r$) $\in$ $PA$ and                          /* identifies $r_1$ and ...  */
04            ($p$,$r$) $\in$ $PA$ and       /* ...  identifies state transition $\delta(q,p)$ */
05            ($\overline{r}$,$a$) $\in$ $APA$ and                                /* identifies $a_1$ */
06            ($\overline{\overline{r'}}$,$a$) $\in$ $APA$                                   /* identifies $r_2$ */
07        then
08            remove-permission-assignment ($p$,$r$)
09            remove-permission-assignment ($q$,$r$)
10            add-permission-assignment ($p'$,$r$)
11            add-permission-assignment ($q'$,$r'$)
```

  It would seem that our definition of $C_{pq}$ is against the spirit of role-based access control which assumes inheritance of permissions by roles. Specifically, if $(p,r) \in PA$ and $r < r'$, then $p$ is implicitly assigned to $r'$. For example, we could have written line 05 as $(q,r) \in {\downarrow}P(r)$. We have chosen not to do this in order to keep the presentation as simple as possible and because the result is unaffected. Note that the ordered triple $(a_\top, r_\top, r_\top)$, for example, does not satisfy the conditional part of $C_{p_1q_1}$ because, although $r_\top$ is implicitly assigned permissions $p_1$ and $q_1$ by inheritance, $a_\top$ is not assigned permission $\overline{r}_\top$ (and hence line 03 evaluates to false). In other words, had we chosen to have conditional commands of the form `if assigned(`$p$`,`$r$`)` rather than `if (`$p$`,`$r$`) in` $PA$, where `assigned(`$p,r$`)` returns true if $p \in {\downarrow}P(r)$, there would still be a unique triple that satisfies the conditional part of $C_{p_1q_1}$.

- The new cell contains $b$

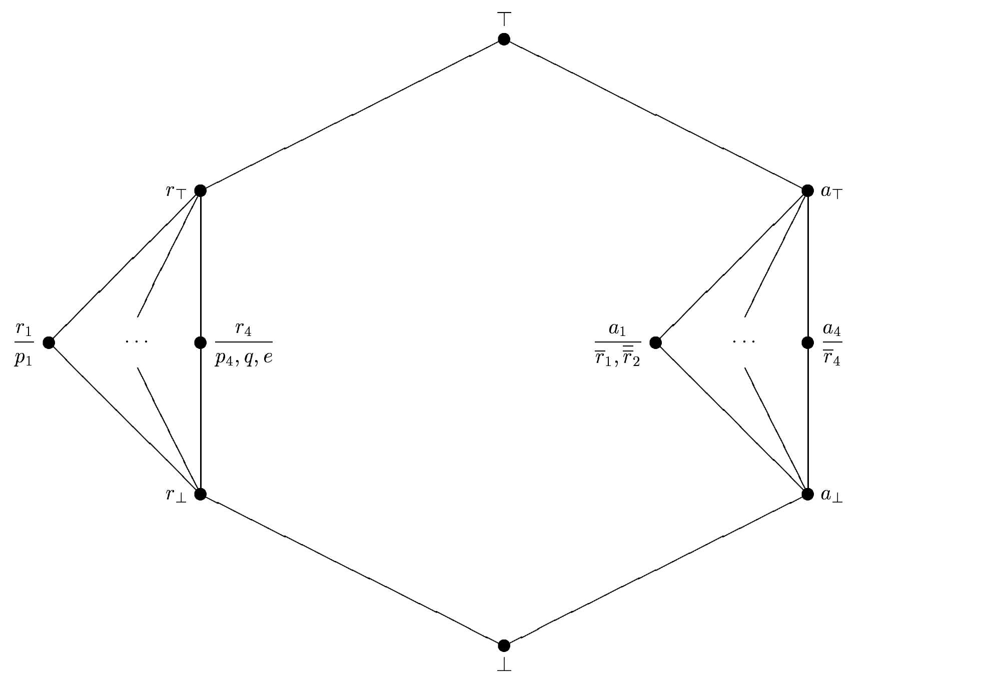
  We define the generic command $D_{pq}$ in Command 5.3. A new role, $r'$, is created by $a_\top$ and the permission $q'$ (corresponding to the new state of the DTM) is assigned to $r'$. Furthermore, a corresponding administrative role, $a'$, is created to manage $r'$ and the appropriate tuples are added to the `can-assignp`, `can-revokep` relations. In addition, $a$ removes $(e,r)$ from $PA$ and adds $(e, r')$ to $PA$. A further administrative role $a''$ is required to amend the permissions of $a$. With reference to Figure 5.3a, and assuming there is a state transition $\delta(q, p_4) = (q', p', 1)$, the logical tests of the corresponding command, $D_{p_4 q}$, are only satisfied by the parameters in the ordered tuple $(a_4, r_4, \_, \_, a_\top)$. (The underscore denotes any valid role identifier, since these parameters refer to the new roles. In Figure 5.3b these are $r_5$ and $a_5$, respectively. Note also that $(a_\top, r_4, \_, \_, a_\top)$ does not satisfy line 07 of the command.) Lines 03 and 04 fix the identity of the ordinary role that satisfies the conditional part of the command. Line 05 establishes that the read-write head is at the rightmost non-blank symbol. Line 06 fixes the administrative role that satisfies the conditional statement since only one administrative role has the permission $\overline{r}$. Line 07 ensures that $a_\top$ is the only role that can be the parameter $a''$.

**Command 5.3**

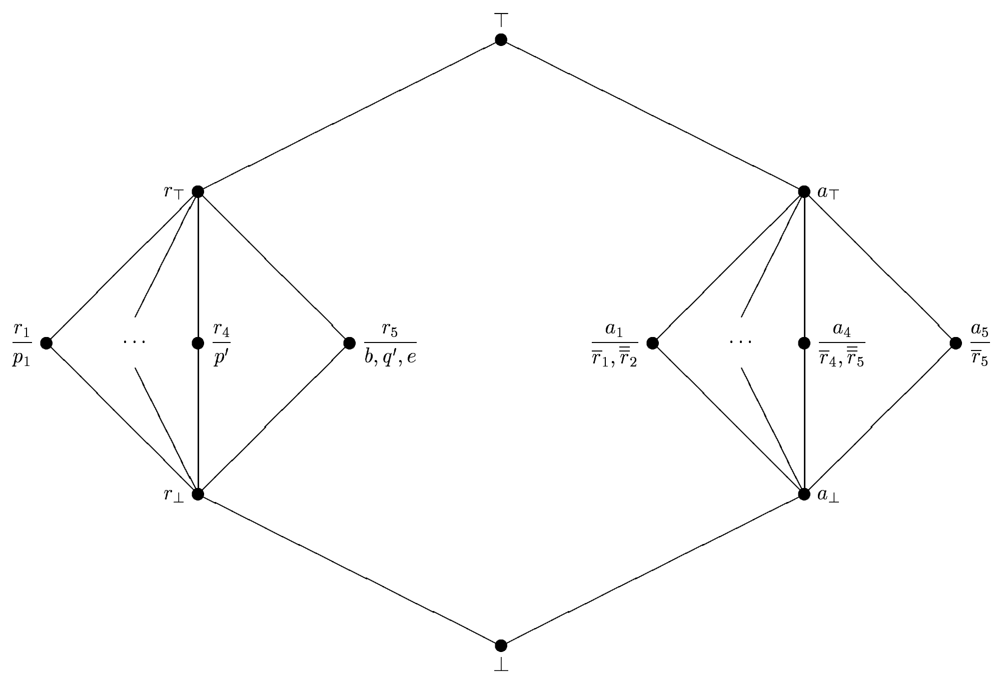
```
01  $D_{pq}$($a$ role, $r$ role, $r'$ role, $a'$ role, $a''$ role)        /* $(a_4, r_4, r_5, a_5, a_\top)$ */
02      if
03          ($q$,$r$) $\in$ $PA$ and                             /* identifies $r_4$ and ...  */
04          ($p$,$r$) $\in$ $PA$ and        /* ...  identifies state transition $\delta(q,p)$ */
05          ($e$,$r$) $\in$ $PA$ and
06          ($\overline{r}$,$a$) $\in$ $APA$ and                                 /* identifies $a_4$ */
07          ($\overline{a}$,$a''$) $\in$ $APA$                                   /* identifies $a_\top$ */
08      then
09          remove-permission-assignment ($p$,$r$)
10          remove-permission-assignment ($q$,$r$)
11          remove-permission-assignment ($e$,$r$)
12          add-role $r'$                                                  /* $r_5$ */
13          add-edge ($r'$,$r_\top$)                                 /* edge ($r_5$,$r_\top$) */
14          add-edge ($r_\bot$,$r'$)                                 /* edge ($r_\bot$,$r_5$) */
15          add-role $a'$                                                  /* $a_5$ */
16          add-edge ($a'$,$a_\top$)                                 /* edge ($a_5$,$a_\top$) */
17          add-edge ($a_\bot$,$a'$)                                 /* edge ($a_\bot$,$a_5$) */
18          add-permission-assignment ($p'$,$r$)
19          add-permission-assignment ($q'$,$r'$)
20          add-permission-assignment ($e$,$r'$)
21          add-permission-assignment ($b$,$r'$)
22          add-permission-assignment ($\overline{\overline{r'}}$,$a$)                     /* ($\overline{\overline{r}}_5$,$a_4$) */
23          add-permission-assignment ($\overline{r'}$,$a'$)                    /* ($\overline{r}_5$,$a_5$) */
24          add-permission-assignment ($\overline{a'}$,$a''$)                   /* ($\overline{a}_5$,$a_\top$) */
```

As in the proof of **SP:HRU**, the logical tests ensure there is a single command which is applicable to the current role hierarchy, while the body of the command preserves the correspondence between the role hierarchy and the DTM.

The case when the state transition function causes the read-write head to move left is simulated in an analogous way. We omit the details. ■



(a) Configuration $p_1p_2p_3qp_4$


$\top$
$r_\top$
$a_\top$
$\frac{r_1}{p_1}$
$\cdots$
$\frac{r_4}{p'}$
$\frac{r_5}{b, q', e}$
$\frac{a_1}{\overline{r}_1, \overline{\overline{r}}_2}$
$\cdots$
$\frac{a_4}{\overline{r}_4, \overline{\overline{r}}_5}$
$\frac{a_5}{\overline{r}_5}$
$r_\perp$
$a_\perp$
$\perp$

(b) Configuration $p_1p_2p_3p'q'b$ after the state transition $\delta(q, p_4) = (q', p', 1)$

**Figure 5.3:** Simulating the read-write head moving right over a blank symbol

### 5.2.2 RBAC96/ARBAC97

In this context, a role-based protection system is parameterized by the set of permissions, the set of commands, the set of RBAC96 relations and the set of ARBAC97 relations. Role insertion in RRA97 requires that the new role has a single parent role and a single child role. Therefore, the operation `add-role` now takes three parameters. Note that $(a_\bot, a_\top)$ and $(r_\bot, r_\top)$ in Figure 5.2 are encapsulated ranges. We also assume the existence of additional primitive operations which add and remove tuples from each of the relations in ARBAC97.

**Theorem 5.2.2** *The safety problem for RBAC96/ARBAC97 is undecidable.*

**Proof** We use the same constructions as in Theorem 5.2.1 and define the following tuples:

$$\begin{aligned}
(a_i, \texttt{true}, [r_i, r_i]) &\in \texttt{can-assignp},\ 1 \leqslant i \leqslant 4;\\
(a_i, [r_i, r_i]) &\in \texttt{can-revokep},\ 1 \leqslant i \leqslant 4;\\
(a_i, \texttt{true}, [r_{i+1}, r_{i+1}]) &\in \texttt{can-assignp},\ 1 \leqslant i \leqslant 3;\\
(a_\top, \texttt{true}, [a_\bot, a_\top)) &\in \texttt{can-assignp};\\
(a_\top, [a_\bot, a_\top)) &\in \texttt{can-revokep};\\
(a_\top, (a_\bot, a_\top)) &\in \texttt{can-modify};\ \text{and}\\
(a_\top, (r_\bot, r_\top)) &\in \texttt{can-modify}.
\end{aligned}$$

Hence $(a_\bot, a_\top)$ and $(r_\bot, r_\top)$ are authority ranges. It can easily be verified that $(a_\bot, a_\top)$ and $(r_\bot, r_\top)$ are create ranges. In particular, new roles can be inserted in the hierarchy with $a_\bot$ and $a_\top$ ($r_\bot$ and $r_\top$) as child and parent nodes, respectively. Each administrative role, $a_x$, can assign and revoke permissions to the role, $r_x$, and the most senior administrative role can assign and revoke permissions to all other administrative roles. Furthermore, the `can-modify` relation contains the tuples $(a_\top, (a_\bot, a_\top))$ and $(a_\top, (r_\bot, r_\top))$. In particular, the administrative role $a_\top$ can create administrative roles in the range $(a_\bot, a_\top)$ and ordinary roles in the range $(r_\bot, r_\top)$. In short, by construction, we guarantee that the commands defined in the proof of Theorem 5.2.1 satisfy the requirements of ARBAC97, and that appropriate conditional statements, such as `is-create-range`$(a_\bot, a_\top)$ can be added to Command 5.3 and will be satisfied by the system.[3]

We also assume that an additional permission $p_\top$, say, is assigned to $a_\top$, enabling $a_\top$ to add and remove tuples from the `can-assignp`, `can-revokep` and `can-modify` relations. Commands of the form $D_{pq}$ need to be modified so that new tuples are added to the ARBAC97 relations. Specifically, we include the additional lines

```
add-can-assignp($a'$,true, $[r, r]$)
add-can-revokep($a$, $[r, r]$)
```

in the pseudo-code given in Command 5.3, where `add-can-assignp` and `add-can-revokep` are additional primitive operations. Note that lines `12` – `14` in Command 5.3 are replaced by the single line `add-role`$(a', a_\bot, a_\top)$. Lines `15` – `17` are replaced by the single line `add-role`$(r', r_\bot, r_\top)$. ■

[3]We will not formally define Boolean functions such as `is-create-range` and rely on the reader's intuition to interpret the obvious intended semantics.

It is not obvious whether the safety problem is undecidable if we omit administrative roles and administrative permissions. Clearly we can use the PRA97 relations to control the assignment and revocation of permissions to roles, but we need a mechanism within ARBAC97 for controlling the addition of tuples to ARBAC97 relations. A cursory analysis suggests that RBAC96 (which includes administrative roles and permissions) is more expressive than RBAC96/ARBAC97 without administrative roles and permissions. This is a matter for further research.

We note in passing that the relationship between the ARBAC97 relations and administrative permissions seems to be analogous to the relationship between the security properties and the protection matrix in the Bell-LaPadula model (Bell and LaPadula 1973a). In particular, the administrative permissions refine the pre-conditions imposed by the ARBAC97 relations on permission- and user-role assignment in a similar way to that of the protection matrix refining authorized access by subjects to objects given the pre-conditions of the simple security property and *-property.

### 5.2.3 RBAC96/SARBAC

An obvious question to ask is whether SARBAC has any impact on **SP:RBAC**. We prove that the question is still undecidable, although the relative simplicity of the analysis leads to some preliminary suggestions for limiting role-based commands so that the safety problem becomes decidable.

**Theorem 5.2.3** ***SP:RBAC*** *is undecidable in the RBAC96/SARBAC model.*

**Proof** (Sketch) We indicate how the role hierarchy simulates a DTM and how the `admin-authority` relation is used. We no longer require a separate administrative role hierarchy so the analysis is simpler than in RBAC96/ARBAC97. Figure 5.4a shows the extended role hierarchy simulating the configuration $q_1p_1p_2p_3p_4$ and corresponds to the hierarchy in Figure 5.2a. Figure 5.4b shows the extended role hierarchy simulating the configuration $p_1p_2p_3p'q'b$ and corresponds to the hierarchy in Figure 5.3b.

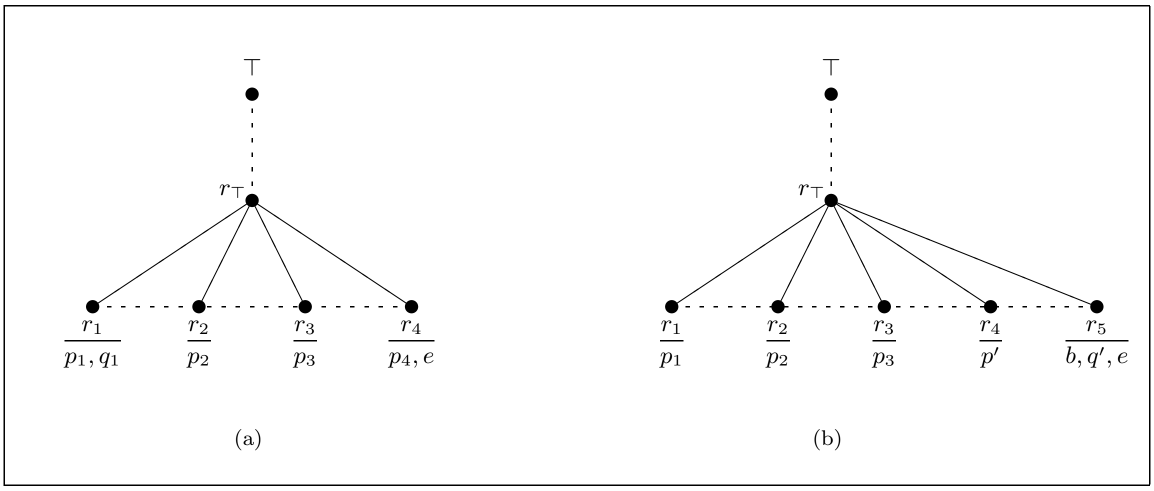


**Figure 5.4:** Extended role hierarchies in RBAC96/SARBAC for **SP:RBAC**

Table 5.2 shows the `admin-authority` relation that gives rise to the the extended hierarchy shown in Figure 5.4a. Rather than show all the tuples in the relation explicitly, we have given the general form for the tuples. Note that the `admin-authority` relation provides a natural way of ordering the roles in the hierarchy so that they correspond with the DTM tape. Note also that $\top$ can add triples to the `admin-authority` relation.

| `admin-authority` | |
|---|---|
| Administrative Role | Role |
| $r_i$ | $r_{i+1}$ |
| $\top$ | $r_\top$ |

Table 5.2: The `admin-authority` relation

Commands 5.4 and 5.5 are analogous to Commands 5.2 and 5.3, respectively. In Command 5.4 line `03` identifies the role corresponding to the square that the read-write head is scanning and line `05` identifies the role to which the new state of the DTM should be assigned. Note that if the read-write head is not over the first square of the tape a command of the form $C_{pq}$ is satisfied by two distinct tuples, $(r_i, r_{i+1}, \top)$ and $(r_i, r_{i+1}, r_{i-1})$. However, the effect of the command is identical in both instances because assignments and revocations in lines `08` – `11` only affect the first two parameters of the command. In short, the construction will still faithfully mimic the behaviour of the DTM. (Similar comments are applicable to Command 5.5.)

**Command 5.4**

```
01  $C_{pq}$($r$ role, $r'$ role, $a$ role)                      /* $(r_1,r_2,\top)$ */
02      if
03          ($q$,$r$) $\in$ $PA$ and                           /* identifies $r_1$ */
04          ($p$,$r$) $\in$ $PA$ and
05          ($r$,$r'$) $\in$ admin-authority and               /* identifies $r_2$ */
06          $r$ $\in$ $\sigma(a)$
07      then
08          remove-permission-assignment ($p$,$r$)
09          remove-permission-assignment ($q$,$r$)
10          add-permission-assignment ($p'$,$r$)
11          add-permission-assignment ($q'$,$r'$)
```

■

### 5.2.4 Discussion

We have shown that if we define arbitrary role-based access control commands, then the safety problem in a role-based access control protection system is undecidable. Clearly, the next stage is to consider restrictions on role-based access control commands that result in decidable and tractable instances of the safety problem. In particular, it would be interesting to find results that correspond to those in Table 5.1, although the complexity of commands in role-based access control means that various additional restrictions are possible.

It is also clear that the operational behaviour of role-based access control can be rather more complicated than that of the protection matrix model. In particular, the conditional statements in

**Command 5.5**

```
$D_{pq}$($r$ role, $r'$ role, $r''$ role, $a$ role)                 /* ($r_4$,$r_5$,$r_\top$,$\top$) */
    if
        ($q$,$r$) $\in$ $PA$ and                                    /* identifies $r_4$ */
        ($p$,$r$) $\in$ $PA$ and
        ($e$,$r$) $\in$ $PA$ and
        ($r$,$r''$) $\in$ $RH$ and                                  /* identifies $r_\top$ */
        $r''$ $\in$ $\sigma(a)$                                     /* identifies $\top$ */
    then
        remove-permission-assignment($p$,$r$)
        remove-permission-assignment($q$,$r$)
        remove-permission-assignment($e$,$r$)
        add-role($r'$,$\emptyset$,$\{r''\}$)                        /* role $r_5$; edge ($r_5$,$r_\top$) */
        add-permission-assignment($p'$,$r$)
        add-permission-assignment($q'$,$r'$)
        add-permission-assignment($e$,$r'$)
        add-permission-assignment($b$,$r'$)
        add-admin-authority($r$,$r'$)                               /* ($r_4$,$r_5$) */
```

commands are more complicated because of the requirements imposed by ARBAC97. Conditional statements also lack the uniformity of those in the protection matrix model.

The interpretation of operational semantics in RBAC96 is complicated by the existence of administrative permissions and an administrative model to control the assignment of users and permissions to roles. It would seem that either mechanism could be used to implement security requirements with regard to assignment of users and permissions to roles. Administrative permissions obviously enable individual administrative roles to assign permissions and users to given roles, and therefore support a fine-grained approach to role assignments. Alternatively, the `can-assign` and `can-assignp` relations offer a more coarse-grained approach to assignment. It is not clear whether or how these two features of RBAC96/ARBAC97 are intended to interact.

It is encouraging that the analysis of the safety problem in Theorem 5.2.3 provides further evidence for the administrative simplicity and clarity of SARBAC. It is clear that the commands will be easier to implement in RBAC96/SARBAC. In particular, the conditional statements need only consider administrative scope, constraint satisfaction and permission assignment. In ARBAC97, there are several different tests that need to be employed: create ranges, encapsulated ranges, authority ranges, immediate authority ranges, membership of ranges, as well as constraint satisfaction and permission assignment.

The proof of the undecidability of **SP:RBAC** suggests that if the width of $R$ can be bounded in some way, then the safety problem will be decidable. This is analogous to restricting the behaviour of the leading diagonal in the protection matrix. This in turn suggests that a role-based protection system can be constructed in which an antichain "computes" solutions to the Post correspondence problem. That is, we conjecture that **SP:RBAC** in a monotonic role-based protection system is also undecidable.

# Chapter 6

# Antichain Completions of a Poset

The material in this chapter is mainly derived from Crampton and Loizou (2000) and Crampton and Loizou (2001b). It provides a rigorous framework for the material on conflict of interest policies in Chapter 7 and a motivation for the secure hierarchical authorization framework in Chapter 8. In fact, it was the development of conflict of interest policies, which are considered to be antichains in a powerset, that led us to consider antichains in the more general setting of arbitrary partial orders and gave rise to the material in this chapter.

As we observed in Lemma 2.1.1, the lattices $\mathcal{I}(X)$ and $\mathcal{F}(X)$ are completions of the poset $X$ under the mappings $x \mapsto \downarrow x$ and $x \mapsto \uparrow x$, respectively. The purpose of this chapter is to prove that if $X$ is finite, then a completion of $X$ using antichains exists. We prove the existence of such a completion by defining two partial orderings, $\preccurlyeq$ and $\preccurlyeq'$, on the set of antichains and showing that the resulting posets, $\langle \mathcal{A}(X), \preccurlyeq \rangle$ and $\langle \mathcal{A}(X), \preccurlyeq' \rangle$, are isomorphic to the lattice of order ideals.

This chapter is arranged as follows. In Section 6.1 we define an ordering $\preccurlyeq$ on $\mathcal{A}(X)$ and prove that the resulting poset is isomorphic to $\mathcal{I}(X)$. We then describe the join and meet operations on $\mathcal{A}(X)$, and present two simple examples. In Section 6.2 we define a second ordering $\preccurlyeq'$ on $\mathcal{A}(X)$ and establish that $\langle \mathcal{A}(X), \preccurlyeq' \rangle$ is also a completion of $\langle X, \leqslant \rangle$. We also summarize the isomorphisms between the alternative completions of $\langle X, \leqslant \rangle$. In Section 6.3 we present some results analogous to well known results for $\mathcal{I}(X)$.

## 6.1 The lattice $\langle \mathcal{A}(X), \preccurlyeq \rangle$

In this section we prove that a completion of $X$ exists in a lattice of antichains. We first define a binary relation $\preccurlyeq$ on $\mathcal{A}(X)$ and prove in Lemma 6.1.1 that it is a partial order. We prove some further elementary preparatory results and then state and prove that $\langle \mathcal{A}(X), \preccurlyeq \rangle$ is isomorphic to $\langle \mathcal{I}(X), \subseteq \rangle$. Examples of $\langle \mathcal{A}(X), \preccurlyeq \rangle$ are shown in Figure 6.3 and Figure 6.5e.

**Definition 6.1.1** *Let $\langle X, \leqslant \rangle$ be a poset. For all $A, B \in \mathcal{A}(X)$, define*

$$A \preccurlyeq B \text{ if, and only if, for all } a \in A, \text{ there exists } b \in B \text{ such that } a \leqslant b.$$

**Lemma 6.1.1** *Let $\langle X, \leqslant \rangle$ be a poset. Then $\langle \mathcal{A}(X), \preccurlyeq \rangle$ is a poset.*

**Proof** Clearly $\preccurlyeq$ is reflexive and transitive. We prove $\preccurlyeq$ is anti-symmetric by contradiction. Suppose that $A \preccurlyeq B$ and $B \preccurlyeq A$, but $A \neq B$. Without loss of generality we can choose $a \in A$ such that $a \notin B$. Since $A \preccurlyeq B$, there exists $b \in B$ such that $a < b$. Furthermore, $b \notin A$ since $A \in \mathcal{A}(X)$ and hence contains no chain. Therefore, there exists $a' \in A$ such that $b < a'$ since $B \preccurlyeq A$. Therefore, we have $a < b < a'$ with $a, a' \in A$, contradicting the fact that $A$ is an antichain. ■

**Remark 6.1.1** *$\langle 2^X, \preccurlyeq \rangle$ is a pre-order, not a partial order, as $\preccurlyeq$ is not anti-symmetric. For example, using the poset of Figure 6.1a, $\{b, e\} \preccurlyeq \{e\}$ and $\{e\} \preccurlyeq \{b, e\}$, but $\{b, e\} \neq \{e\}$.*

**Lemma 6.1.2** *Let $\overline{\alpha} : 2^X \to \mathcal{A}(X)$ and $\underline{\alpha} : 2^X \to \mathcal{A}(X)$, where*

$$\overline{\alpha}(Y) = \{y \in Y : y \text{ is a maximal element in } Y\},$$
$$\underline{\alpha}(Y) = \{y \in Y : y \text{ is a minimal element in } Y\}.$$

*Then*

$$\overline{\alpha}(Y) \subseteq Y, \tag{6.1}$$
$$\text{for all } y \in Y, \text{ there exists } y' \in \overline{\alpha}(Y) \text{ such that } y \leqslant y', \tag{6.2}$$
$$\overline{\alpha}(Y) \in \mathcal{A}(X), \tag{6.3}$$
$$\overline{\alpha} \text{ is well defined;} \tag{6.4}$$

*and*

$$\underline{\alpha}(Y) \subseteq Y, \tag{6.5}$$
$$\text{for all } y \in Y, \text{ there exists } y' \in \underline{\alpha}(Y) \text{ such that } y' \leqslant y, \tag{6.6}$$
$$\underline{\alpha}(Y) \in \mathcal{A}(X), \tag{6.7}$$
$$\underline{\alpha} \text{ is well defined.} \tag{6.8}$$

**Proof**

*Proof of* (6.1) Suppose $y \in \overline{\alpha}(Y)$. Then, by definition, $y \in Y$. The result follows. □

*Proof of* (6.2) Recall that $X$ is finite, by assumption. Let $y \in Y$. If $y \in \overline{\alpha}(Y)$, then $y \leqslant y$ and the result follows. Otherwise, there exists $y_1 \in Y$ such that $y < y_1$. If $y_1 \in \overline{\alpha}(Y)$, then $y \leqslant y_1$ and the result follows. Applying the above argument repeatedly we can construct a chain $y < y_1 < \cdots < y_n$ in $Y$. This process necessarily terminates, since $Y \subseteq X$ is finite. The result follows. □

*Proof of* (6.3) By (6.2), if $Y \neq \emptyset$, then $\overline{\alpha}(Y) \neq \emptyset$. Let $x, y \in \overline{\alpha}(Y)$ such that $y \leqslant x$. Since $y$ is a maximal element $x = y$. That is, for all $x, y \in \overline{\alpha}(Y)$, either $x = y$ or $x \parallel y$. The result follows. □

*Proof of* (6.4) Suppose $\overline{\alpha}(Y) = A_1$ and $\overline{\alpha}(Y) = A_2$ with $A_1 \neq A_2$. Then without loss of generality there exists $x \in A_1 \setminus A_2$. Furthermore, since $A_1 \subseteq Y$ there exists $x' \in A_2$ such that

$x \leqslant x'$ by (6.2). Since $A_1$ is an antichain $x' \notin A_1$. Similarly, $A_2 \subseteq Y$; hence there exists $x'' \in A_1$ such that $x' \leqslant x''$. By transitivity we have $x \leqslant x' \leqslant x''$ with $x, x'' \in A_1$, which is a contradiction as $A_1$ is an antichain. $\square$

The results in (6.5) – (6.8) can be proved in an analogous way to (6.1) – (6.4) and are omitted. $\blacksquare$

To simplify the notation in the remainder of this chapter we will write $\overline{Y}$ to denote $\overline{\alpha}(Y)$ and $\underline{Y}$ to denote $\underline{\alpha}(Y)$.

**Lemma 6.1.3** *Let $\phi : \mathcal{I}(X) \to \mathcal{A}(X)$ and $\psi : \mathcal{A}(X) \to \mathcal{I}(X)$ such that*

$$\phi(I) = \overline{I} \quad \textit{and} \quad \psi(A) = {\downarrow}A.$$

*Then $\phi$ and $\psi$ are bijections. Moreover $\phi = \psi^{-1}$.*

**Proof** Since $X$ is assumed to be finite, $\mathcal{I}(X)$ is finite, and hence for $\phi$ to be a bijection it suffices to show that $\phi$ is one-to-one. Suppose then that $\phi(I) = \phi(J) = A$, where $I, J \in \mathcal{I}(X)$ and $A \in \mathcal{A}(X)$. For all $i \in I$, there exists $a \in A$ such that $a \geqslant i$ by Lemma 6.1.2. Furthermore, $a \in J$, since $A \subseteq J$. $J$ is an ideal, $i \in X$, $a \in J$ and $a \geqslant i$. Therefore, by definition, $i \in J$. That is, $I \subseteq J$. Similarly $J \subseteq I$ and hence $I = J$. That is, $\phi$ is one-to-one.

We first prove by contradiction that $\psi$ is well defined. Suppose that $\psi(A) = I_1$, $\psi(A) = I_2$ and $I_1 \neq I_2$. Then without loss of generality there exists $i \in I_1$ such that $i \notin I_2$. Hence $i \notin A$ (since $A \subseteq I_2$). Therefore, by definition of $\psi$, there exists $a \in A$ such that $i < a$. Hence we have $a \in I_2$, $i \notin I_2$ with $i < a$. That is, $I_2$ is not an ideal.

Finally, we show that $\overline{{\downarrow}A} = A$. That is, $\phi(\psi(A)) = A$. Suppose that $a \in A$. Then $a \in {\downarrow}A$. Now $a \in \overline{{\downarrow}A}$ (otherwise there exists $b \in {\downarrow}A$ such that $b > a$, which implies there exists $a' \in A$ such that $a' \geqslant b > a$ and hence that $A$ is not antichain). That is, $A \subseteq \overline{{\downarrow}A}$. Suppose that $a \in \overline{{\downarrow}A}$. Then $a \in {\downarrow}A$ and hence there exists $a' \in A$ such that $a \leqslant a'$. Now $a' \in {\downarrow}A$, hence $a = a'$, otherwise $a \notin \overline{{\downarrow}A}$. That is, $\overline{{\downarrow}A} \subseteq A$; hence $\overline{{\downarrow}A} = A$ and therefore $\phi$ and $\psi$ are mutually inverse bijections. $\blacksquare$

**Corollary 6.1.1** $|\mathcal{I}(X)| = |\mathcal{A}(X)| = |\mathcal{F}(X)|$.

**Proof** The fact that $|\mathcal{I}(X)| = |\mathcal{A}(X)|$ follows immediately from Lemma 6.1.3. Clearly $\zeta : \mathcal{I}(X) \to \mathcal{F}(X)$, where $\zeta(I) = X \setminus I$ is a bijection. Hence $|\mathcal{I}(X)| = |\mathcal{F}(X)|$. $\blacksquare$

Figure 6.1 illustrates the one-to-one correspondence between antichains, order ideals and order filters. Figure 6.1a shows an antichain in the lattice in Figure 2.1a. Figures 6.1b and 6.1c show the corresponding order filter and ideal, respectively. In particular, the antichain $\{d, e\}$ is the set of minimal elements in the order filter and the set of maximal elements in the order ideal.

**Theorem 6.1.1** *$\langle \mathcal{A}(X), \preccurlyeq \rangle$ is isomorphic to the lattice $\langle \mathcal{I}(X), \subseteq \rangle$. Furthermore, $\mathcal{A}(X)$ is a completion of $X$.*

**Proof** We prove that the functions $\phi : \mathcal{I}(X) \to \mathcal{A}(X)$ and $\psi : \mathcal{A}(X) \to \mathcal{I}(X)$, where $\phi(I) = \overline{I}$ and $\psi(A) = {\downarrow}A$, are order-preserving.

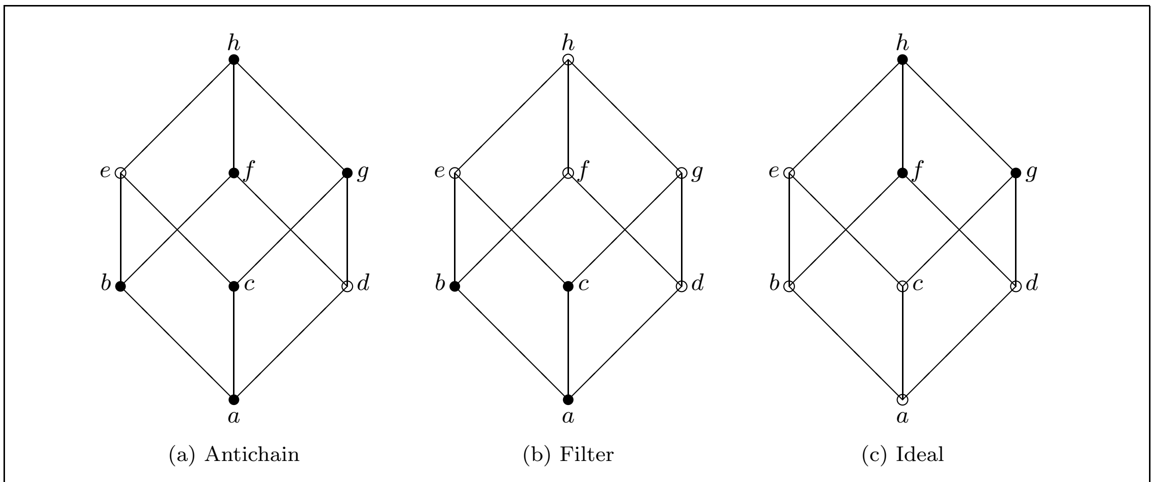


**Figure 6.1:** The correspondence between antichains, order filters and order ideals

- $\phi$ is order-preserving – that is, for all $I, J \in \mathcal{I}(X)$,

$$I \subseteq J \text{ implies } \overline{I} \preccurlyeq \overline{J}. \tag{6.9}$$

*Proof of* (6.9) Suppose $I \subseteq J$. Then $\overline{I} \subseteq I \subseteq J$. Hence, if $i \in \overline{I}$ then $i \in J$. Therefore, by (6.2), there exists $j \in \overline{J}$ such that $i \leqslant j$. That is, $\overline{I} \preccurlyeq \overline{J}$. □

- $\psi$ is order-preserving – that is, for all $A, B \in \mathcal{A}(X)$,

$$A \preccurlyeq B \text{ implies } {\downarrow}A \subseteq {\downarrow}B. \tag{6.10}$$

*Proof of* (6.10) Suppose $A \preccurlyeq B$ and $a \in {\downarrow}A$. Then there exists $a' \in A$ such that $a \leqslant a'$. Since $A \preccurlyeq B$ there exists $b \in B$ such that $a \leqslant a' \leqslant b$. Hence $a \in {\downarrow}B$. That is, ${\downarrow}A \subseteq {\downarrow}B$. □

Since $\phi$ and $\psi$ are mutually inverse order-preserving bijections, the first part of the result now follows by Theorem 2.1.1, while the second part follows immediately from Lemma 2.1.1 and Definition 6.1.1, where $x \mapsto \{x\}$ is the required order-embedding. ■

We now consider the binary operations on the lattice $\langle \mathcal{A}(X), \preccurlyeq \rangle$ which are explicitly described by the following lemma.

**Lemma 6.1.4** *For all $A, B \in \mathcal{A}(X)$, $A \wedge B = \overline{{\downarrow}A \cap {\downarrow}B}$ and $A \vee B = \overline{A \cup B}$.*

**Proof** We first make the observation that, by construction, $\overline{{\downarrow}A \cap {\downarrow}B}$, $\overline{A \cup B} \in \mathcal{A}(X)$.

- $\overline{{\downarrow}A \cap {\downarrow}B}$ is a lower bound of $A$ and $B$.

*Proof* Suppose $x \in \overline{{\downarrow}A \cap {\downarrow}B}$. Then by (6.1) $x \in {\downarrow}A \cap {\downarrow}B$, and therefore $x \in {\downarrow}A$ and $x \in {\downarrow}B$. Hence there exists $a \in A$ such that $x \leqslant a$ and there exists $b \in B$ such that $x \leqslant b$. Therefore, $\overline{{\downarrow}A \cap {\downarrow}B} \preccurlyeq A$ and $\overline{{\downarrow}A \cap {\downarrow}B} \preccurlyeq B$. □

- $\overline{A \cup B}$ is an upper bound of $A$ and $B$.

*Proof* Suppose $a \in A$. Then $a \in A \cup B$ and hence there exists $a' \in \overline{A \cup B}$ such that $a \leqslant a'$. Therefore, $A \preccurlyeq \overline{A \cup B}$. Similarly $B \preccurlyeq \overline{A \cup B}$. □

- $\overline{\downarrow A \cap \downarrow B}$ is the greatest lower bound of $A$ and $B$.

  *Proof* Suppose $C \in \mathcal{A}(X)$ such that $C \preccurlyeq A$ and $C \preccurlyeq B$. Then, by Proposition 2.1.1 and (6.10), $C \subseteq \downarrow C \subseteq \downarrow A$, $C \subseteq \downarrow C \subseteq \downarrow B$, and therefore $C \subseteq \downarrow A \cap \downarrow B$. Hence, by (6.9), and since $C \in \mathcal{A}(X)$, $C = \overline{C} \preccurlyeq \overline{\downarrow A \cap \downarrow B}$. □

- $\overline{A \cup B}$ is the least upper bound of $A$ and $B$.

  *Proof* Suppose $C \in \mathcal{A}(X)$ such that $A \preccurlyeq C$ and $B \preccurlyeq C$. Then $A \subseteq \downarrow A \subseteq \downarrow C$ and $B \subseteq \downarrow B \subseteq \downarrow C$. Therefore $A \cup B \subseteq \downarrow C$ and, by (6.9), $\overline{A \cup B} \preccurlyeq \overline{\downarrow C} = C$. □

■

## 6.2 The lattice $\langle \mathcal{A}(X), \preccurlyeq' \rangle$

We first state two results analogous to Lemma 6.1.1 and Lemma 6.1.3.

**Lemma 6.2.1** *Let $\langle X, \leqslant \rangle$ be a poset. For all $A, B \in \mathcal{A}(X)$, define*

$$A \preccurlyeq' B \text{ if, and only if, for all } b \in B \text{ there exists } a \in A \text{ such that } a \leqslant b.$$

*Then $\langle \mathcal{A}(X), \preccurlyeq' \rangle$ is a poset.*

**Proof** Clearly $\preccurlyeq'$ is reflexive and transitive. We prove $\preccurlyeq'$ is anti-symmetric by contradiction. Suppose $A, B \in \mathcal{A}(X)$ and $A \preccurlyeq' B$, $B \preccurlyeq' A$, but $A \neq B$. Without loss of generality we can choose $a \in A$ such that $a \notin B$. Since $B \preccurlyeq' A$, there exists $b \in B$ such that $b < a$. Furthermore, $b \notin A$ since $A \in \mathcal{A}(X)$ and hence contains no chain. Therefore, there exists $a' \in A$ such that $a' < b$ since $A \preccurlyeq' B$. Therefore, we have $a' < b < a$ with $a, a' \in A$, but, since $A \in \mathcal{A}(X)$ we have a contradiction. ■

**Lemma 6.2.2** *Let $\phi' : \mathcal{F}(X) \to \mathcal{A}(X)$ and $\psi' : \mathcal{A}(X) \to \mathcal{F}(X)$ such that*

$$\phi'(F) = \underline{F} \quad \text{and} \quad \psi'(A) = \uparrow A.$$

*Then $\phi'$ and $\psi'$ are mutually inverse bijections.*

**Proof** The result is proved in an analogous way to Lemma 6.1.3 and is omitted. ■

**Theorem 6.2.1** *$\langle \mathcal{A}(X), \preccurlyeq' \rangle$ is isomorphic to $\langle \mathcal{I}(X), \subseteq \rangle$.*

**Proof** We prove the functions $\phi' : \mathcal{F}(X) \to \mathcal{A}(X)$ and $\psi' : \mathcal{A}(X) \to \mathcal{F}(X)$, where $\phi'(F) = \underline{F}$ and $\psi'(A) = \uparrow A$ are order-preserving.

- $\phi'$ is order-preserving – that is, for $F, G \in \mathcal{F}(X)$,

$$F \supseteq G \text{ implies } \underline{F} \preccurlyeq' \underline{G} \tag{6.11}$$

*Proof* Suppose $F \supseteq G$. Then $F \supseteq G \supseteq \underline{G}$. Hence, if $g \in \underline{G}$ then $g \in F$. Therefore, there exists $f \in \underline{F}$ such that $f \leqslant g$. That is, $\underline{F} \preccurlyeq' \underline{G}$. □

- $\psi'$ is order-preserving – that is, for $A, B \in \mathcal{A}(X)$,

$$A \preccurlyeq' B \text{ implies } {\uparrow}A \supseteq {\uparrow}B \tag{6.12}$$

*Proof* Suppose $A \preccurlyeq' B$ and $b \in {\uparrow}B$. Then there exists $b' \in B$ such that $b' \leqslant b$. Since $A \preccurlyeq' B$ there exists $a \in A$ such that $a \leqslant b'$. Therefore, $a \leqslant b$. Hence $b \in {\uparrow}A$. That is ${\uparrow}A \supseteq {\uparrow}B$. □

Since $\phi'$ and $\psi'$ are mutually inverse order-preserving bijections, we have, by Theorem 2.1.1, that $\langle \mathcal{A}(X), \preccurlyeq' \rangle \cong \langle \mathcal{F}(X), \supseteq \rangle$. Hence, since $\langle \mathcal{I}(X), \subseteq \rangle \cong \langle \mathcal{F}(X), \supseteq \rangle$ via the mapping $I \mapsto X \setminus I$, we have $\langle \mathcal{A}(X), \preccurlyeq' \rangle \cong \langle \mathcal{I}(X), \subseteq \rangle$ via the mapping $A \mapsto X \setminus {\uparrow}A$. ■

In Figure 6.2 we summarize the relationships between the lattices $\langle \mathcal{I}(X), \subseteq \rangle$, $\langle \mathcal{F}(X), \supseteq \rangle$, $\langle \mathcal{A}(X), \preccurlyeq \rangle$ and $\langle \mathcal{A}(X), \preccurlyeq' \rangle$. Figure 6.3 shows the lattices $\langle \mathcal{A}(2^{[3]}), \preccurlyeq \rangle$ and $\langle \mathcal{A}(2^{[3]}), \preccurlyeq' \rangle$.

$$\begin{tikzcd}
\langle \mathcal{I}(X), \subseteq \rangle \arrow[r, "I \mapsto X \setminus I"] \arrow[d, "I \mapsto \overline{I}"'] & \langle \mathcal{F}(X), \supseteq \rangle \arrow[d, "F \mapsto \underline{F}"] \\
\langle \mathcal{A}(X), \preccurlyeq \rangle & \langle \mathcal{A}(X), \preccurlyeq' \rangle \arrow[l, "A \mapsto \overline{X \setminus {\uparrow}A}"]
\end{tikzcd}$$

**Figure 6.2:** The isomorphisms between $\langle \mathcal{I}(X), \subseteq \rangle$, $\langle \mathcal{F}(X), \supseteq \rangle$, $\langle \mathcal{A}(X), \preccurlyeq \rangle$ and $\langle \mathcal{A}(X), \preccurlyeq' \rangle$

**Lemma 6.2.3** *For all $A, B \in \mathcal{A}(X)$, $A \wedge' B = \underline{A \cup B}$ and $A \vee' B = \underline{{\uparrow}A \cap {\uparrow}B}$.*

**Proof**

- $\underline{A \cup B}$ is a lower bound of $A$ and $B$.

*Proof* Suppose $a \in A$. Then $a \in A \cup B$, and hence there exists $x \in \underline{A \cup B}$ such that $x \leqslant a$. Therefore, $\underline{A \cup B} \preccurlyeq' A$. Similarly $\underline{A \cup B} \preccurlyeq' B$. □

- $\underline{{\uparrow}A \cup {\uparrow}B}$ is an upper bound of $A$ and $B$.

*Proof* Suppose $x \in \underline{{\uparrow}A \cap {\uparrow}B}$. Then $x \in {\uparrow}A \cap {\uparrow}B$ and hence $x \in {\uparrow}A$ and $x \in {\uparrow}B$. Therefore, there exists $a \in A$ and $b \in B$ such that $a \leqslant x$ and $b \leqslant x$. Therefore, $A \preccurlyeq' \underline{{\uparrow}A \cap {\uparrow}B}$ and $B \preccurlyeq' \underline{{\uparrow}A \cap {\uparrow}B}$. □

- $\underline{A \cup B}$ is the greatest lower bound of $A$ and $B$.

  *Proof* Suppose $C \in \mathcal{A}(X)$ such that $C \preccurlyeq' A$ and $C \preccurlyeq' B$. Then, by (6.12), $\uparrow C \supseteq A$ and $\uparrow C \supseteq B$. Therefore, $A \cup B \subseteq \uparrow C$ and, by (6.11), $C = \underline{\uparrow C} \preccurlyeq' \underline{A \cup B}$. □

- $\underline{\uparrow A \cap \uparrow B}$ is the least upper bound of $A$ and $B$.

  *Proof* Suppose $C \in \mathcal{A}(X)$ such that $A \preccurlyeq' C$ and $B \preccurlyeq' C$. Then, by (6.12), $\uparrow A \supseteq C$ and $\uparrow B \supseteq C$. Hence $C \subseteq \uparrow A \cap \uparrow B$ and, by (6.11), $\underline{\uparrow A \cap \uparrow B} \preccurlyeq' \underline{C} = C$. □

■

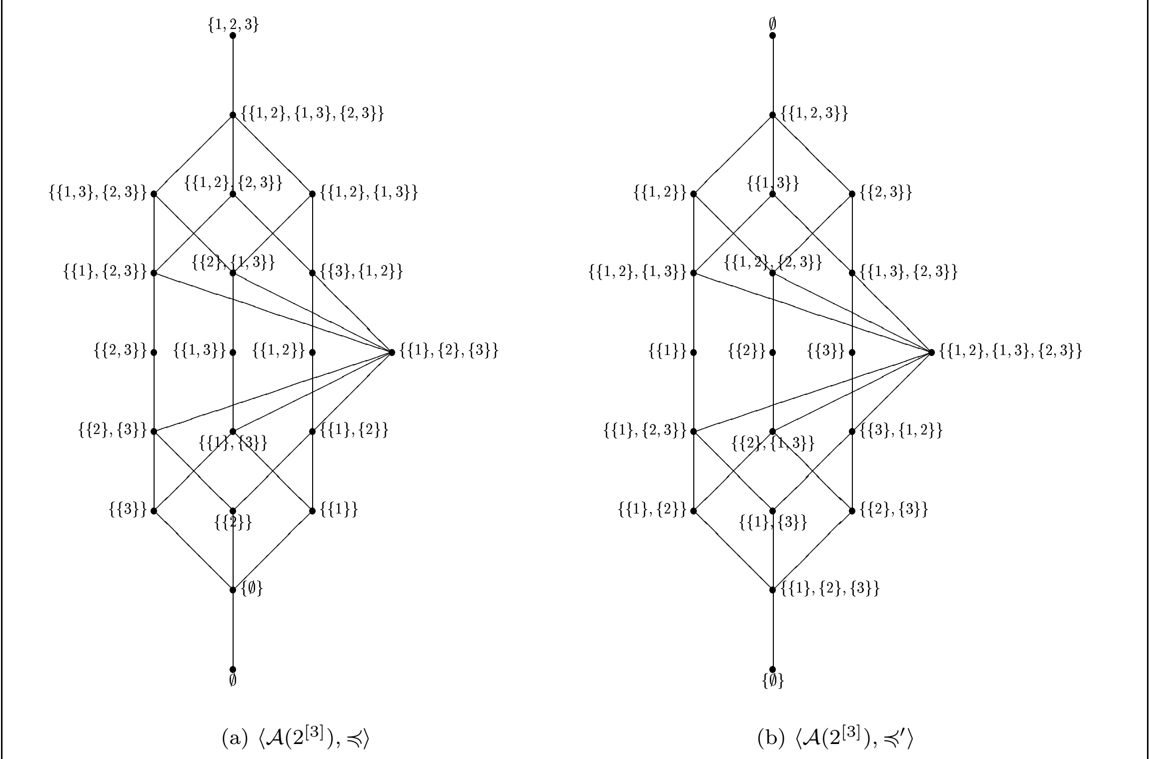


**Figure 6.3:** The lattices $\langle \mathcal{A}(2^{[3]}), \preccurlyeq \rangle$ and $\langle \mathcal{A}(2^{[3]}), \preccurlyeq' \rangle$

## 6.3 Further results

This section presents further results for $\langle \mathcal{A}(X), \preccurlyeq \rangle$ which are analogous to standard results for $\langle \mathcal{I}(X), \subseteq \rangle$. Results corresponding to Propositions 6.3.1, 6.3.2 and 6.3.3 can be found in Davey and Priestley (1990), labelled therein as Theorems 8.17, 2.31 and 8.22, respectively.

**Proposition 6.3.1** *Let $L$ be a finite distributive lattice. Then $L$ is isomorphic to $\mathcal{A}(\mathcal{J}(L))$, where $\mathcal{J}(L)$ is the set of join-irreducible elements in $L$.*

**Proof** Consider the function $\phi : L \to \mathcal{A}(\mathcal{J}(L))$, where

$$\phi(x) = \overline{\mathcal{J}(L) \cap {\downarrow}x}.$$

We first prove that $\phi$ is an order-embedding.

$$\begin{aligned}
x \leqslant y &\Rightarrow {\downarrow}x \subseteq {\downarrow}y \\
&\Rightarrow \mathcal{J}(L) \cap {\downarrow}x \subseteq \mathcal{J}(L) \cap {\downarrow}y \\
&\Rightarrow \overline{\mathcal{J}(L) \cap {\downarrow}x} \preccurlyeq \overline{\mathcal{J}(L) \cap {\downarrow}y} \quad \text{by (6.9)} \\
&\Rightarrow \phi(x) \preccurlyeq \phi(y) \\
\phi(x) \preccurlyeq \phi(y) &\Rightarrow \overline{\mathcal{J}(L) \cap {\downarrow}x} \preccurlyeq \overline{\mathcal{J}(L) \cap {\downarrow}y} \\
&\Rightarrow {\downarrow}\overline{\mathcal{J}(L) \cap {\downarrow}x} \subseteq {\downarrow}\overline{\mathcal{J}(L) \cap {\downarrow}y} \quad \text{by (6.10)} \\
&\Rightarrow \mathcal{J}(L) \cap {\downarrow}x \subseteq \mathcal{J}(L) \cap {\downarrow}y \quad \text{by Lemma 6.1.3} \\
&\Rightarrow {\downarrow}x \subseteq {\downarrow}y \\
&\Rightarrow x \leqslant y
\end{aligned}$$

It remains to prove that $\phi$ is a bijection. Suppose that $\phi(x) = \phi(y)$. Then we have

$$\begin{aligned}
\mathcal{J}(L) \cap {\downarrow}x = \mathcal{J}(L) \cap {\downarrow}y &\Rightarrow {\downarrow}x = {\downarrow}y \\
&\Rightarrow {\downarrow}x \subseteq {\downarrow}y \quad \text{and} \quad {\downarrow}y \subseteq {\downarrow}x \\
&\Rightarrow x \leqslant y \quad \text{and} \quad y \leqslant x \\
&\Rightarrow x = y.
\end{aligned}$$

Hence $\phi$ is one-to-one.

We now prove that $\phi$ is onto. Let $A \in \mathcal{A}(\mathcal{J}(L))$, where $A = \{a_1, \ldots, a_k\}$. Define $a = a_1 \vee \cdots \vee a_k$. We claim that $A = \phi(a)$. Let $x \in A$. Then $x = a_i$ for some $i$. Hence $x \in \mathcal{J}(L)$ is join-irreducible and $x \leqslant a$. Hence $x \in \phi(a)$. That is, $A \subseteq \phi(a)$. Let $x \in \phi(a)$. Then $x \leqslant a = a_1 \vee \cdots \vee a_k$. Since $L$ is distributive, $x \leqslant a_i$ for some $i$, by Lemma 2.1.2, and since $A$ is an antichain, $x = a_i$. That is, $x \in A$ and hence $\phi(a) \subseteq A$. The result now follows. ■

Figure 6.4 illustrates Proposition 6.3.1, and compares $\mathcal{A}(\mathcal{J}(L))$ and $\mathcal{I}(\mathcal{J}(L))$. The join-irreducible elements are highlighted in Figure 6.4a. It can be seen that $x \in \mathcal{J}(L)$ is mapped to the set $\{x\}$ by $\phi$, which means the construction of $\mathcal{A}(\mathcal{J}(L))$ is more straightforward than that of $\mathcal{I}(\mathcal{J}(L))$.

Of course, Proposition 6.3.1 is merely a re-statement of Birkhoff's Representation Theorem for finite distributive lattices (Birkhoff 1933), and as such can be proved as a corollary of that result and Theorem 6.1.1.

We now state without proof two further propositions: the first states the existence of a Dedekind-MacNeille-style completion (MacNeille 1937) using antichains and follows from the fact that any element of the Dedekind-MacNeille completion is an ideal; the second states several results regarding standard constructions on posets and is a direct translation of the associated results for ideals.

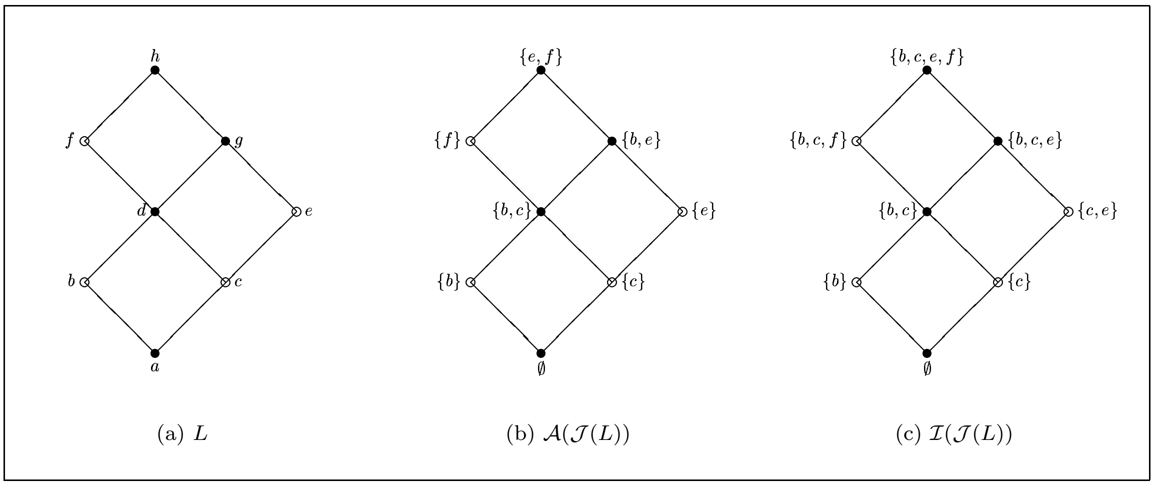


**Figure 6.4:** $L$, $\mathcal{A}(\mathcal{J}(L))$ and $\mathcal{I}(\mathcal{J}(L))$: $\mathcal{J}(L) = \{b, c, e, f\}$

**Proposition 6.3.2** *The lattice $\langle DM_{\mathcal{A}}(X), \preccurlyeq\rangle$, where $DM_{\mathcal{A}}(X) = \{\overline{Y} : Y \subseteq X,\ Y^{ul} = Y\}$, is a completion of $X$ via the order-embedding $\phi : X \hookrightarrow DM_{\mathcal{A}}(X)$ such that $\phi(x) = \{x\}$. Furthermore, $DM_{\mathcal{A}}(X)$ is isomorphic to the Dedekind-MacNeille completion of $X$, $DM(X)$.*

**Proposition 6.3.3** *Let $X$ be a finite poset. Then*

- $\mathcal{A}(X)^{\partial} \cong \mathcal{A}(X^{\partial})$, *where $X^{\partial}$ is the dual of $X$;*
- $\mathcal{A}(\perp \oplus X) \cong \emptyset \oplus (\{\perp\} \oplus \mathcal{A}(X))$, *where $\oplus$ is the linear sum of two posets and $\perp$ denotes a bottom element;*
- $\mathcal{A}(X \oplus \top) \cong \mathcal{A}(X) \oplus \{\top\}$, *where $\top$ denotes a top element;*
- $\mathcal{A}(X_1 \dot\cup X_2) \cong \mathcal{A}(X_1) \times \mathcal{A}(X_2)$, *where $\dot\cup$ denotes disjoint union and $\times$ denotes cartesian product.*

Figure 6.5 shows several different completions of a given poset. Notice that $X = \{a, b\} \dot\cup \{c, d, e, f\}$. The completion in Figure 6.5e can be seen to be the direct product of the lattices of antichains for each of these two sets. In particular, the long diagonals represent $\mathcal{A}(\{a, b\}) = \{\emptyset, \{a\}, \{b\}\}$, where $\emptyset \preccurlyeq \{a\} \preccurlyeq \{b\}$.

## 6.4 Summary and discussion

We have shown that given a finite poset $X$, it is possible to define two different orderings on the set of antichains in $X$ and that the resulting posets are isomorphic to $\langle \mathcal{I}(X), \subseteq\rangle$. Hence the set of antichains in a poset forms a lattice which, in addition, is a completion of the poset. Furthermore, many known results for $\langle \mathcal{I}(X), \subseteq\rangle$ can be adapted to produce corresponding results for $\langle \mathcal{A}(X), \preccurlyeq\rangle$. In particular, there exists a Dedekind-MacNeille-style completion $\langle DM_{\mathcal{A}}(X), \preccurlyeq\rangle$ with $DM_{\mathcal{A}}(X) \subseteq \mathcal{A}(X)$, and an analogue of Birkhoff's Representation Theorem for finite distributive lattices.

We believe that, in general, it is easier to determine the elements of $\mathcal{A}(X)$ than those of $\mathcal{I}(X)$, and hence to determine the structure of $\mathcal{I}(X)$ via the isomorphism. In particular, every singleton subset of $X$ is an antichain, and no element of $\mathcal{A}(X)$ has more elements than the width of $X$.

We hope to investigate whether an algorithm to compute $\mathcal{A}(X)$ exists which necessarily has lower time and space complexity than one to compute $\mathcal{I}(X)$. (It is easy to see that a description of $\mathcal{A}(X)$ requires less space than $\mathcal{I}(X)$.)

One difficulty with the Dedekind-MacNeille completion is that the characterization of the elements in the lattice is difficult to visualize and awkward to compute. We believe that a simpler characterization of the elements of $DM_{\mathcal{A}}(X)$ may be possible and hence that a simple characterization of the elements in the Dedekind-MacNeille completion may be found.

In the next two chapters we explore two applications of $\mathcal{A}(X)$. The first application, described in the following chapter, is a simple model for access control policies which include separation of duty policies as a special case. We model such policies as antichains in a suitable powerset and show that the join operation associated with the $\preccurlyeq'$ ordering can be used to combine two policies to create a policy that implements the requirements of both policies.

The second application, described in Chapter 8, is to provide the theoretical justification for the secure hierarchical authorization framework. This model in fact is more of a template that can be used to define access control models which support features analogous to the simple security and *-properties of the Bell-LaPadula model. The development of the secure hierarchical authorization framework was inspired by the observation that the security lattice in the Bell-LaPadula model can be regarded as a lattice of antichains in a particular poset. In the secure hierarchical authorization framework we associate each entity with an antichain in a hierarchy of positions. The lattice $\langle \mathcal{A}(X), \preccurlyeq \rangle$ guarantees that for any set of antichains we can always find a more senior antichain.

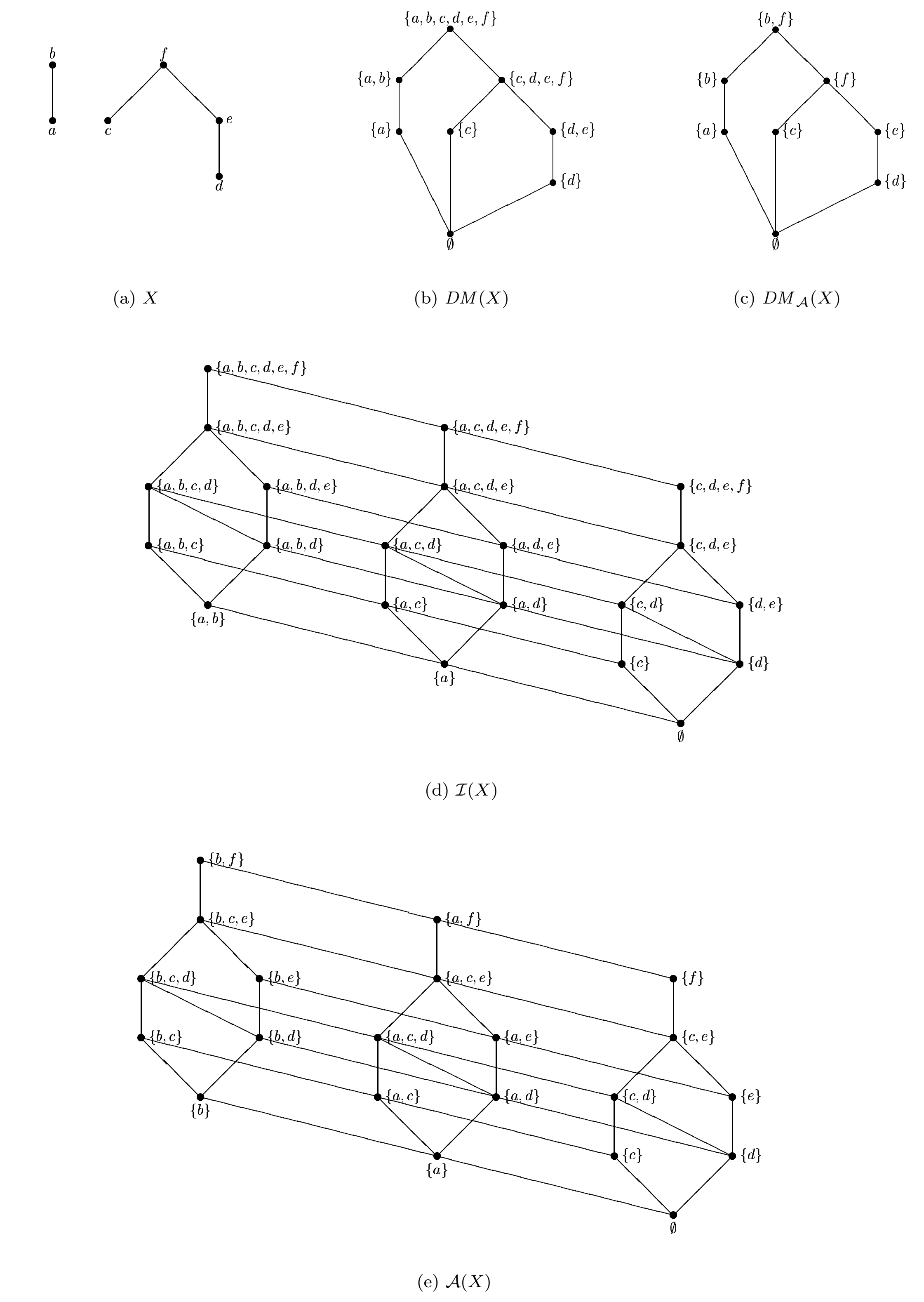


**Figure 6.5:** Completions of the poset in Figure 2.1c

# Chapter 7

# Conflict of Interest Policies

In this chapter we present a set-based approach to separation of duty which has the benefit of great simplicity. A consequence of this simplicity is that we are able to consider the complexity of several aspects of separation of duty policies. The material in this chapter arose from our work on the specification of access control policies (Crampton et al. 1999) and because we felt that existing approaches suffered from a lack of generality and were over-elaborate. The main contribution of this chapter is to demonstrate that our model is sufficiently expressive to model separation of duty policies, while being sufficiently simple to permit a detailed analysis of separation of duty in general.

O'Shea (1997) developed a logic of access control and a Prolog implementation that could reason about the consequences of a particular configuration of file permissions in a UNIX system. The motivation for this was to provide a logical framework for software tools that could assist a system administrator to configure access control lists. We extended this work by conducting experiments in which we specified simple access control policies and testing whether a given implementation of security on UNIX and Windows NT platforms met the specification (Crampton et al. 1999; Crampton et al. 2001).

In this work we modelled access control policies as subsets or elements of a set $X$. For example, an *authorization-based policy*, $P^+ \subseteq X$, specifies which elements of $X$ are authorized. Intuitively, $P^+$ is violated if there exists $x$ such that the access control mechanism permits $x$ and $x \notin P^+$. Similarly, a *prohibition-based policy*, $P^- \subseteq X$, specifies which elements of $X$ are prohibited. Intuitively, $P^-$ is violated if there exists $x$ such that the access control mechanism permits $x$ and $x \in P^-$. (It is obvious that for every policy $P^+$ there is an equivalent policy $P^- = X \setminus P^+$.) Such policies can be used to articulate confidentiality requirements (a user must not read a certain file, say) and integrity requirements (a user must not write to a certain file, say).

However, when one considers separation of duty policies, it becomes clear that each component of the policy (which we refer to as a *constraint*) must be a subset of $X$. Specifically, a constraint states those elements of $X$ which form a sensitive combination. Hence a separation of duty policy, $\mathcal{P}$, can be represented as a set of subsets of $X$. The material in this chapter is derived from this simple observation.

We will demonstrate that for suitable choices of $X$ and constraints, several interesting policies arise which define situations that conflict with the interests of the enterprise but are not separation of duty policies in the usual sense. Hence we will use the term *conflict of interest policy* throughout this chapter.

In the next section we introduce our formal model of conflict of interest policies, and the circumstances under which such policies are violated. We define a conflict of interest policy and the canonical representation of a conflict of interest policy. We show that the canonical representation is formally equivalent to a Sperner family. A preliminary version of this section appeared in Crampton and Loizou (2001a). In Section 7.2 we illustrate the application of the formal model to the protection matrix model and a role-based access control model. In the latter context, we show how conflict of interest policies can be used to impose a ceiling on the roles to which a user can be assigned. In Section 7.3 we discuss the structural complexity of the formal model. In particular, we derive an upper bound for the length of a canonical conflict of interest policy. We also derive upper and lower bounds for the number of canonical conflict of interest policies (and hence the number of Sperner families). In order to establish our upper bound we define the novel concept of a *bi-symmetric chain partition.* Finally, in Section 7.5 we consider a simplification to the formal model and summarize the work of the chapter.

## 7.1 Formal model

Let $X$ be a set defined in an access control model. We will refer to $X$ as an *access control context* (or simply *context*). For example, $X$ may be the set of all possible triples in the HRU model.

An *access control environment* (or simply *environment*) $E$ is a subset of $X$. For example, in the HRU model, $E$ is the set of triples encoded by the access control matrix.

**Definition 7.1.1** *A* conflict of interest constraint *(or simply* constraint*) is a subset of $X$. A* conflict of interest policy *(or simply* policy*) is a set of conflict of interest constraints.*

*An environment $E$* satisfies *a conflict of interest policy $\mathcal{P}$ if, and only if, for all $P \in \mathcal{P}$, $P \cap E \neq P$ (that is, $P \not\subseteq E$), and* violates *it otherwise. We denote the set of environments which satisfy $\mathcal{P}$ by $\mathcal{E}(\mathcal{P})$.*

In other words, a conflict of interest policy states which subsets of $X$ cannot be present simultaneously in the environment, and is satisfied provided the environment does not include any conflict of interest constraint in the policy. Table 7.1 shows three conflict of interest policies

$$\mathcal{P}_1 = \{\{1,2\},\{2,3\}\}, \quad \mathcal{P}_2 = \{\{1\},\{2,3\}\}, \quad \mathcal{P}_3 = \{\{1\},\{1,2\},\{2,3\}\},$$

and the environments which satisfy (ticked) and violate (crossed) each policy (when $n = 3$). We make the following observations about Definition 7.1.1.

- A singleton set $\{x\} \in \mathcal{P}$, implies that $x \in X$ is prohibited from entering the environment $E$. Specifically, the policy

$$P^- = \{x_1, \ldots, x_n\}$$

can be expressed as the conflict of interest policy

$$\mathcal{P} = \{\{x_1\}, \dots, \{x_n\}\}.$$

Hence our framework can accommodate prohibition policies which articulate confidentiality and integrity constraints, as well as separation of duty constraints.

- If $\mathcal{P} = \{\emptyset\}$ then no environment satisfies $\mathcal{P}$ (since $\emptyset \subseteq E$ for all $E \subseteq X$).
- If $\mathcal{P} = \emptyset$ then every environment satisfies $\mathcal{P}$ (since $\mathcal{P}$ contains no constraints).

Henceforth we assume that the access control context $X$ is a finite set and hence can be identified with $[n]$, where $n = |X|$. (We are justified in doing this, because any finite set, $X = \{x_1, \dots, x_n\}$, can be identified with the set $[n]$ via the bijective mapping $x_i \mapsto i$, $1 \leqslant i \leqslant n$.) We will denote $\mathcal{A}(2^{[n]})$ by $\mathcal{A}_n$.

| **Environment** | **Policy** | | |
|---|---|---|---|
| | $\mathcal{P}_1 = \{\{1,2\},\{2,3\}\}$ | $\mathcal{P}_2 = \{\{1\},\{2,3\}\}$ | $\mathcal{P}_3 = \{\{1\},\{1,2\},\{2,3\}\}$ |
| $\emptyset$ | ✓ | ✓ | ✓ |
| $\{1\}$ | ✓ | ✗ | ✗ |
| $\{2\}$ | ✓ | ✓ | ✓ |
| $\{3\}$ | ✓ | ✓ | ✓ |
| $\{1,2\}$ | ✗ | ✗ | ✗ |
| $\{1,3\}$ | ✓ | ✗ | ✗ |
| $\{2,3\}$ | ✗ | ✗ | ✗ |
| $\{1,2,3\}$ | ✗ | ✗ | ✗ |

Table 7.1: A comparison of conflict of interest policies and environments

**Definition 7.1.2** *Given two conflict of interest policies, $\mathcal{P}, \mathcal{Q}$, we say $\mathcal{P}$ is* weaker than *(or* less restrictive than *or* is enforced by*) $\mathcal{Q}$ if $\mathcal{E}(\mathcal{P}) \supset \mathcal{E}(\mathcal{Q})$; we will also say $\mathcal{Q}$ is* stronger *(or* more restrictive than *or* enforces*) $\mathcal{P}$. $\mathcal{P}$ and $\mathcal{Q}$ are* equivalent *if $\mathcal{E}(\mathcal{P}) = \mathcal{E}(\mathcal{Q})$.*

In Table 7.1, $\mathcal{P}_1$ is weaker than $\mathcal{P}_2$, for example. From Table 7.1 we also see that $\mathcal{P}_2$ and $\mathcal{P}_3$ are equivalent. In fact we have the following result.

**Proposition 7.1.1** *Suppose $\mathcal{P} \in 2^{2^{[n]}}$, $P_1 \subset P_2$ for some $P_1, P_2 \in \mathcal{P}$ and $\mathcal{P}' = \mathcal{P} \setminus \{P_2\}$. Then $\mathcal{E}(\mathcal{P}) = \mathcal{E}(\mathcal{P}')$. In other words, $\mathcal{P}$ and $\mathcal{P}'$ are equivalent.*

**Proof** We prove the equivalent statement that an environment $E$ satisfies $\mathcal{P}$ if, and only if, $E$ satisfies $\mathcal{P}'$.

$\Rightarrow$ It follows immediately from the fact that $\mathcal{P}' \subset \mathcal{P}$.

$\Leftarrow$ The proof proceeds by contradiction. Suppose, then, that $E$ satisfies $\mathcal{P}'$ but does not satisfy $\mathcal{P}$. Clearly $P_2 \subseteq E$ is the only possible way in which $E$ does not satisfy $\mathcal{P}$. However, by construction, $P_1 \subset P_2 \subseteq E$, and hence $E$ does not satisfy $\mathcal{P}'$.

■

Given that $\langle 2^{2^{[n]}}, \subseteq\rangle$ is a poset, and using the result of Proposition 7.1.1, we propose the following definition of a canonical representation of a conflict of interest policy.

**Definition 7.1.3** *Given a conflict of interest policy $\mathcal{P} \in 2^{2^{[n]}}$, we define the* canonical representation of $\mathcal{P}$ *to be $\underline{\mathcal{P}} \in \mathcal{A}_n$, where $\underline{\mathcal{P}}$ is the set of minimal elements in $\mathcal{P}$. A policy $\mathcal{Q} \in \mathcal{A}_n$ is called a* canonical conflict of interest policy.

In other words, given a conflict of interest policy, its canonical representation is obtained by removing all conflict of interest constraints which are a superset of another constraint in the policy. By Proposition 7.1.1, the canonical representation of a policy is equivalent to the original policy (and by (6.8), it is unique). For example $\mathcal{P}_2$ is the canonical representation of $\mathcal{P}_3$ in Table 7.1.

Henceforth, therefore, we assume that all policies are in their canonical form. In other words, given an access control context $X$, the set of canonical conflict of interest policies is $\mathcal{A}_n$, where $n = |X|$, and every canonical conflict of interest policy is a Sperner family. Given that $\langle 2^{[n]}, \subseteq\rangle$ is a partial order, we have the following corollary of Theorem 6.2.1.

**Proposition 7.1.2** *For all $\mathcal{P}, \mathcal{Q} \in \mathcal{A}_n$, define*

$$\mathcal{P} \preccurlyeq' \mathcal{Q} \text{ if, and only if, for all } Q \in \mathcal{Q} \text{ there exists } P \in \mathcal{P} \text{ such that } P \subseteq Q.$$

*Then $\langle \mathcal{A}_n, \preccurlyeq'\rangle$ is a complete lattice.*

The following proposition demonstrates that the formal definition of $\preccurlyeq'$ corresponds exactly to the intuitive definition of strength given in Definition 7.1.2.

**Proposition 7.1.3** *For all $\mathcal{P}, \mathcal{Q} \in \mathcal{A}_n$, $\mathcal{P} \preccurlyeq' \mathcal{Q}$ if, and only if, $\mathcal{P}$ is stronger than $\mathcal{Q}$.*

**Proof** The proof in both directions proceeds by contradiction.

- $\Rightarrow$ Given $\mathcal{P} \preccurlyeq' \mathcal{Q}$, suppose $\mathcal{E}(\mathcal{P}) \not\subseteq \mathcal{E}(\mathcal{Q})$. Then there exists $E \in \mathcal{E}(\mathcal{P})$ such that $E \notin \mathcal{E}(\mathcal{Q})$. Hence there exists $Q \in \mathcal{Q}$ such that $Q \subseteq E$. Since, by assumption, $\mathcal{P} \preccurlyeq' \mathcal{Q}$, for all $Q \in \mathcal{Q}$, there exists $P \in \mathcal{P}$ such that $P \subseteq Q$, and hence we have $P \subseteq Q \subseteq E$. That is, $E \notin \mathcal{E}(\mathcal{P})$, which is a contradiction.

- $\Leftarrow$ Given $\mathcal{P}$ is stronger than $\mathcal{Q}$, suppose $\mathcal{P} \not\preccurlyeq' \mathcal{Q}$. Then, by definition, for some $Q \in \mathcal{Q}$ and for all $P \in \mathcal{P}$, $P \not\subseteq Q$. In other words, for all $P \in \mathcal{P}$, $P \cap Q \subset P$. Therefore, by definition, $Q \in \mathcal{E}(\mathcal{P})$, and, since $\mathcal{P}$ is stronger than $\mathcal{Q}$, $\mathcal{E}(\mathcal{P}) \subseteq \mathcal{E}(\mathcal{Q})$. That is, $Q$ satisfies the policy $\mathcal{Q}$. This is a contradiction since $Q \in \mathcal{Q}$.

■

Since $\langle \mathcal{A}_n, \preccurlyeq'\rangle$ is a complete lattice, the meet and join of any set of policies exists. In particular for any pair of policies, $\mathcal{P}$ and $\mathcal{Q}$, we can find a third policy, $\mathcal{P} \wedge \mathcal{Q}$, the weakest policy which is at least as strong as both $\mathcal{P}$ and $\mathcal{Q}$. In other words, we have a natural definition of composition of policies. Recalling Lemma 6.2.3 we have the following definition.

**Definition 7.1.4** *For all $\mathcal{P}, \mathcal{Q} \in \mathcal{A}_n$, define*

$$\mathcal{P} \wedge \mathcal{Q} = \underline{\mathcal{P} \cup \mathcal{Q}}, \text{ and } \mathcal{P} \vee \mathcal{Q} = \underline{\uparrow\mathcal{P} \cap \uparrow\mathcal{Q}}.$$

**Example 7.1.1** Consider Figure 6.3b. We have, for example,

$$\{\{1\},\{2,3\}\} < \{\{1,2\},\{2,3\}\} < \{\{1,2,3\}\},$$

and by Lemma 6.2.1, or by inspection of Figure 6.3b, it is easily verified that

$$\begin{aligned} \{\{1\},\{2,3\}\} \wedge \{\{2\},\{1,3\}\} &= \{\{1\},\{2\}\}, \\ \{\{1\},\{2,3\}\} \vee \{\{2\},\{1,3\}\} &= \{\{1,2\},\{1,3\},\{2,3\}\}. \end{aligned}$$

## 7.2 Examples and applications

In this section we demonstrate the generic nature of our approach by applying it to two different access control models. In the first set of examples we use the protection matrix model, and in the second a role-based access control model. In both examples we indicate the sets which correspond to $X$ and $E$. We conclude the section with a brief comparison of our approach and existing approaches to separation of duty in role-based access control.

### 7.2.1 The protection matrix model

Let $M$ denote the protection matrix, $O$ the set of objects, $S$ the set of subjects and $A$ the set of access modes. In this case $X = O \times S \times A$ and (for static conflict of interest policies) $E$ is the set of triples encoded by $M$. (The environment in the dynamic case is the set of active triples which have been invoked by subjects and granted by the access control mechanism.)

Let $o_1, o_2 \in O$, $S = \{s_1, \ldots, s_n\}$ and $x \in A$, where $x$ denotes "execute" access. We now give some simple examples of conflict of interest policies.

- Subject $s_1$ is prohibited from executing $o_1$.

  $$\mathcal{P}_1 = \{\{(o_1, s_1, x)\}\}$$

  $\mathcal{P}_1$ is satisfied provided $x \notin [s_1, o_1]$. This is a trivial example of a negative authorization policy.

- No subject can execute both $o_1$ and $o_2$.

  $$\mathcal{P}_2 = \{\{(o_1, s, x), (o_2, s, x)\} : s \in S\}$$

  $\mathcal{P}_2$ is satisfied provided $x \notin ([s, o_1] \cap [s, o_2])$ for all $s \in S$. This is a trivial example of a separation of duty policy.

- There is no "super-user".

$$\mathcal{P}_3 = \bigcup_{i=1}^{n} \{O \times \{s_i\} \times A\}$$

  $\mathcal{P}_3$ is satisfied provided no subject has every access mode to every object.

- No subject is permitted to execute any file.

$$\mathcal{P}_4 = \{\{(o, s, x)\} : o \in O, s \in S\}$$

  $\mathcal{P}_4$ is satisfied if for all $o \in O$ and for all $s \in S$, $x \notin [s, o]$.

- If we combine the features of $\mathcal{P}_1$ and $\mathcal{P}_2$ we see that

$$\mathcal{P}_1 \wedge \mathcal{P}_2 = \mathcal{P}_1 \cup \mathcal{P}_2 \setminus \{(o_1, s_1, x), (o_2, s_1, x)\}$$

  since $\{(o_1, s_1, x)\} \subseteq \{(o_1, s_1, x), (o_2, s_1, x)\}$ (see Proposition 7.1.1).

### 7.2.2 The role-based access control model

We assume the existence of a set of roles $R$, a set of users $U$, and a user-role assignment relation $UA$. We first consider the situation when $X = R$. A static conflict of interest policy gives rise to an environment for each user, namely $\downarrow R(u)$. (A dynamic conflict of interest policy gives rise to an environment for each session, namely $\downarrow R(s)$.)

We now give some typical examples of simple policies. We base these policies on examples and material in Chapter 3.

- No user can be assigned to `MaxRole`. This is a simple role exclusion policy. (Recall that in the role graph model it is unlikely that any user should be assigned to `MaxRole`.)

$$\mathcal{Q}_1 = \{\{\texttt{MaxRole}\}\}$$

  In fact, this policy can be implemented in URA97 by ensuring that `MaxRole` does not appear in any range in the `can-assign` relation, although we would argue that $\mathcal{Q}_1$ is a more natural means of specifying this requirement.

  This type of policy could also be used when a role has been "de-commissioned" and should no longer be used (as in role de-activation in RRA97, for example).

- No user can be assigned to both `PE1` and `QE1`. This is an example of a typical separation of duty constraint in role-based access control.

$$\mathcal{Q}_2 = \{\{\texttt{PE1}, \texttt{QE1}\}\}$$

We now briefly consider the case when $X = U \times R$ which significantly expands the expressive power of conflict of interest policies. In this case, the environment for static conflict of interest policies is $E = \{(u, r) : (u, r') \in UA, r \leqslant r'\}$.

- dave cannot be assigned to role PL1. This type of policy imposes a ceiling on the roles to which dave can be assigned. To our knowledge, such requirements, rather surprisingly in our view, have not received any attention in the literature.

  $$\mathcal{Q}_3 = \{\{(\texttt{dave}, \texttt{PL1})\}\}$$

  We observed in Chapter 4 that the approach adopted in ARBAC97, OASIS and SARBAC does not permit the specification of such policies because user-role assignments are controlled by some restriction on assignment (URA97 constraints, role activation rules and SARBAC constraints, respectively) which applies to all users. Hence $(\texttt{DSO}, \texttt{ED} \wedge \overline{\texttt{PL1}}, [\texttt{PL2}, \texttt{PL2}]) \in$ can-assign, for example, can be used to implement separation of duty, because it applies to all users. However, it is not obvious how the can-assign relation could be used to implement $\mathcal{Q}_3$.)

  We note the following useful application of such a policy. We recall that the role-based access control model is policy neutral (Sandhu et al. 1996), and that it is of considerable value to demonstrate that such a model can be used to simulate mandatory and discretionary access control models (Nyanchama and Osborn 1995; Osborn et al. 2000; Sandhu and Munawer 1998a). It has been convincingly shown that role-based access control can indeed simulate mandatory access control (Osborn et al. 2000) by modelling the security lattice $L$ as two distinct read and write role hierarchies $L_R$ and $L_W$, respectively, where $L_R$ is *isomorphic* to $L$ and $L_W$ is the dual of $L_R$.

  However, we believe the constraints introduced in Osborn et al. (2000) to enforce the information flow policy that is an integral part of the mandatory access control model are rather complicated. We suggest that to achieve this we can simply define a role exclusion policy of a similar form to $\mathcal{Q}_3$ for each user $u$. Figure 7.1 shows a security lattice for the security labels

  $$\texttt{unclassified} < \texttt{classified} < \texttt{secret} < \texttt{top secret}$$

  which we will abbreviate to u, c, s, and t, respectively, and two security categories, $k_1$ and $k_2$. If a user $u$ has security clearance $(\texttt{c}, \{k_1\})$, the conflict of interest policy

  $$\mathcal{Q}_4 = \{\{(u, (\texttt{s}, \{k_1\}))\}, \{(u, (\texttt{c}, \{k_2\}))\}\}$$

  preserves the information flow policy defined by the lattice by preventing $u$ being assigned to, and hence activating, any roles other than $(\texttt{u}, \emptyset)$ and $(\texttt{c}, \{k_1\})$. (Of course, in a role-based access control implementation there would actually be a read and a write lattice, but the example policy can be extended in the obvious way to accommodate this.)

- dave cannot be assigned to both PE1 and QE1. This is a separation of duty policy at the user level. It also implies that dave cannot be assigned to any role in $\uparrow\texttt{PE1} \cap \uparrow\texttt{QE1}$.

  $$\mathcal{Q}_5 = \{\{(\texttt{dave}, \texttt{PE1}), (\texttt{dave}, \texttt{QE1})\}\}$$

- Users $u_1$ and $u_2$ cannot occupy both or one of each of the two roles $r_1$ and $r_2$. This kind of policy was identified in Ahn and Sandhu (2000) and aims to prevent collusion between two (or more) individuals to compromise system security.

$$\mathcal{Q}_6 = \{\{(u_1, r_1), (u_1, r_2)\}, \{(u_2, r_1), (u_2, r_2)\}, \{(u_1, r_1), (u_2, r_2)\}, \{(u_1, r_2), (u_2, r_1)\}\}$$

The first two constraints are simple separation of duty constraints for each of the users, while the other two constraints prevent collusion by the two users.

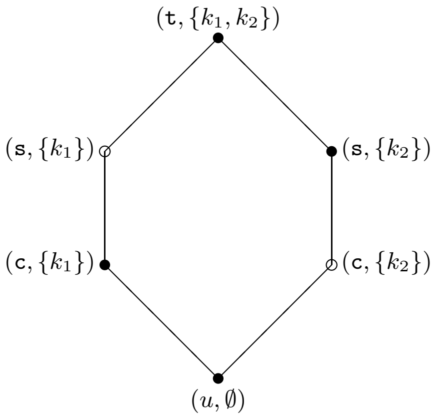


**Figure 7.1:** A security lattice

**Remark 7.2.1** *Note that in Definition 7.1.1 we assumed nothing about the set $X$. However, if $X$ is a partially ordered set, then, in general, a conflict of interest constraint should be defined to be an antichain in $X$ rather than a subset of $X$.*

*For example, consider the role hierarchy in Figure 7.2 and the policy $\mathcal{P} = \{\{r_1, r_3\}, \{r_2, r_3\}\}$. In our framework, the policy $\mathcal{P}$ is reduced to the policy $\mathcal{P}' = \{\{r_1\}, \{r_2, r_3\}\}$.*

*In general, therefore, a conflict of interest policy in a role-based access control model is a member of $\mathcal{A}(\mathcal{A}(R))$. In other words, the constraints of a conflict of interest policy are antichains in $\langle \mathcal{A}(R), \subseteq \rangle$ rather than in $\langle 2^R, \subseteq \rangle$; see Figure 7.2, for example. (In the case of an unordered set $X$ – that is, the order relation is the empty set – the set of antichains is simply $2^X$.)*

## 7.2.3 Comparison with existing models

We now consider the two most significant existing approaches to separation of duty in role-based access control, the NIST model and RCL 2000, and briefly compare them to our approach.

### The NIST model

The most detailed discussion of separation of duty constraints and their implementation in a role-based access control system is found in Gavrila and Barkley (1998) (which refines and outlines an implementation of the NIST model described in Ferraiolo et al. (1995)). The RBAC database includes two binary, irreflexive, symmetric relations `ssd` and `dsd` modelling static and dynamic

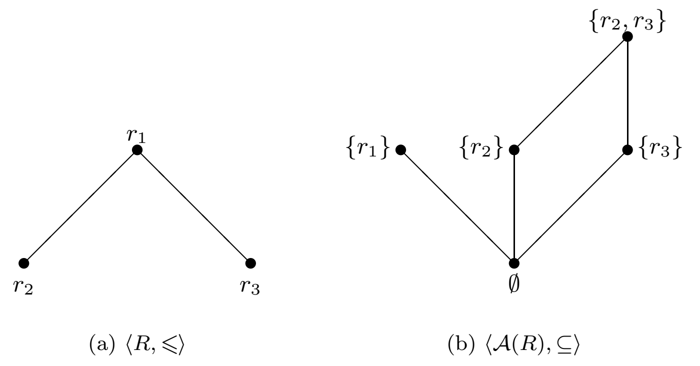

$r_1$
$r_2$
$r_3$
(a) $\langle R, \leqslant \rangle$
$\{r_2, r_3\}$
$\{r_1\}$
$\{r_2\}$
$\{r_3\}$
$\emptyset$
(b) $\langle \mathcal{A}(R), \subseteq \rangle$

**Figure 7.2:** The role hierarchy and conflict of interest policies

separation of duty, respectively. A pair $(r_1, r_2) \in \texttt{ssd}$ is, in our terminology, a conflict of interest constraint. A conflict of interest policy corresponds to the set ssd and is violated if $\{r_1, r_2\} \subseteq R(u)$ for some $u \in U$. In short, the NIST model only considers constraints containing precisely two roles.

We now discuss the additional requirements, identified in Gavrila and Barkley (1998), which ssd (and dsd) must satisfy in a role-based context.

- The ssd relation must be irreflexive.

  The irreflexivity condition is introduced to prevent a mutually exclusive pair $(r, r)$ from being entered into the ssd relation, the assumption being that such a pair would only have the meaning that no user could be assigned to the role $r$. We would argue that there is a useful place for such constraints, particularly when one includes a user component (as in the policy $\{\{(\texttt{dave}, \texttt{PL1})\}\}$, for example).

- The ssd relation must be symmetric.

  The symmetric condition is introduced in order to establish certain logical equivalences between constraints in the NIST model in the presence of a role hierarchy, and to thereby reduce the number of logical tests in the implementation of the database update operations.

- If $(r_1, r_2) \in \texttt{ssd}$ then $\{r_1, r_2\} \in \mathcal{A}(R)$.

  This requirement is a natural counterpart to the irreflexive condition since, if $r_1 \leqslant r_2$ and $(r_1, r_2) \in \texttt{ssd}$, then no user can be assigned to $r_2$ or any role senior to it. We would have no objection to a policy that did not permit any user to be assigned to $r_2$, although, as we observed in Remark 7.2.1 we do require that a conflict of interest constraint be an antichain.

- If $(r_1, r_2) \in \texttt{ssd}$ then $\uparrow r_1 \cap \uparrow r_2 = \emptyset$.

  The justification for this condition is that if $r \in \uparrow r_1 \cap \uparrow r_2$ then no user can be assigned to the role $r$. We do not, in general, wish to impose such a condition on conflict of interest policies, particularly when users are included in the policies (as in $\{\{(\texttt{dave}, \texttt{PE1}), (\texttt{dave}, \texttt{QE1})\}\}$, for example).

In short, we believe the NIST approach (and the broadly similar approach adopted in the role graph model) to separation of duty policies is rather limited.

**RCL 2000**

The role constraint authorization language RCL 2000 supports a flexible and expressive treatment of separation of duty policies. It is described in detail in Ahn and Sandhu (2000); we described its basic features in Chapter 3. We will not attempt a formal comparison of the expressive power of RCL 2000 and our approach. Instead we will demonstrate that examples of RCL 2000 expressions taken from the literature can be easily expressed using Sperner families.

The RCL 2000 expression

$$|\texttt{roles}(\texttt{OE}(U) \cap \texttt{OE}(CR))| \leqslant 1, \tag{7.1}$$

is interpreted in the following way: for the collection of sets of roles, $CR$, no user (that is an element of the set $U$) can be assigned more than one role in any of the sets contained in $CR$. In our terminology, (7.1) states the conditions for satisfaction of the static conflict of interest policy $CR$. Therefore, we would argue that we could simply express $CR$ as a conflict of interest policy $\mathcal{P}$ in which each constraint is a pair (by (7.1) a user cannot be assigned more than one role in any conflicting set). Similarly (3.3) states the conditions for the satisfaction of the dynamic conflict of interest policy $CR$.

In short, we believe that there is little difference in expressive power between RCL 2000 and our approach. However, we believe that the conversion of RCL 2000 expressions to RFOPL expressions is unnecessarily complicated and obscures the fact that separation of duty is essentially a simple concept. In the remainder of this chapter we exploit the simplicity of our approach to establish results on the structural complexity of separation of duty policies.

## 7.3 Structural complexity

The theoretical results of the chapter are contained in this section. The main contributions of this section are to establish an upper bound for the length of a description of a conflict of interest policy, and to improve on the upper and lower bounds for $|\mathcal{A}_n|$ due to Hansel (page 28).

**Definition 7.3.1** *The function $\Sigma : \mathcal{A}_n \to \mathbb{N}$ is defined as follows:*

$$\Sigma(\mathcal{P}) = \begin{cases} 0 & \mathcal{P} = \emptyset, \\ \displaystyle\sum_{P \in \mathcal{P}} |P| & \textit{otherwise.} \end{cases}$$

In other words, $\Sigma(\mathcal{P})$ is a measure of the complexity of describing $\mathcal{P}$ (as a string, for example). The following lemma establishes an upper bound for $\Sigma(\mathcal{P})$ and when the upper bound is attained. It is similar in both statement and proof to Sperner's Theorem (page 27).

**Lemma 7.3.1** *For all $\mathcal{P} \in \mathcal{A}_n$,*

$$\Sigma(\mathcal{P}) \leqslant \lceil n/2 \rceil \binom{n}{\lceil n/2 \rceil},$$

*with equality when*

$$\mathcal{P} = \begin{cases} \{P \subseteq [n] : |P| = \frac{n}{2}\} \quad \textit{or} \quad \{P \subseteq [n] : |P| = \frac{n+2}{2}\} & n \textit{ even,} \\ \{P \subseteq [n] : |P| = \frac{n+1}{2}\} & n \textit{ odd.} \end{cases} \tag{7.2}$$

**Proof** We first note that $\mathcal{P}$ as defined in (7.2)

- belongs to $\mathcal{A}_n$ by construction; and
- $\Sigma(\mathcal{P}) = \lceil n/2 \rceil \binom{n}{\lceil n/2 \rceil}$.

  This is obvious when $n$ is odd. Note that for $0 \leqslant r < n$,

  $$(n-r)\binom{n}{r} = (r+1)\binom{n}{r+1}, \tag{7.3}$$

  and that when $n$ is even $\lceil n/2 \rceil$= n/2. Hence, if $n$ is even, on substituting $r = n/2$ into (7.3) we obtain

  $$\frac{n}{2}\binom{n}{n/2} = (n/2+1)\binom{n}{n/2+1} = \left(\frac{n+2}{2}\right)\binom{n}{(n+2)/2}.$$

We now follow the approach of the original proof of Sperner's Theorem. Suppose $\mathcal{Q} \in \mathcal{A}_n$ and $\Sigma(\mathcal{Q})$ is maximal. We will prove that $\mathcal{Q}$ must be equal to $\mathcal{P}$ as given by (7.2). In order to do this, we show that unless $\mathcal{Q}$ satisfies certain conditions, we can construct two policies, $\mathcal{Q}'$ and $\mathcal{Q}''$, such that $\Sigma(\mathcal{Q}') \geqslant \Sigma(\mathcal{Q})$ and $\Sigma(\mathcal{Q}'') \geqslant \Sigma(\mathcal{Q})$. Let $u = \max_{Q \in \mathcal{Q}} |Q|$, and define

$$\begin{aligned} \mathcal{Q}_u &= \{Q \in \mathcal{Q} : |Q| = u\}, \\ \mathcal{Q}_{u-1} &= \{Q \subseteq [n] : \text{there exists } Q' \in \mathcal{Q}_u \text{ such that } Q \lessdot Q'\}, \\ \mathcal{Q}' &= (\mathcal{Q} \setminus \mathcal{Q}_u) \cup \mathcal{Q}_{u-1}. \end{aligned}$$

In other words, $\mathcal{Q}'$ is formed from $\mathcal{Q}$ by removing the set of elements of maximal cardinality, namely $\mathcal{Q}_u$, and replacing it with $\mathcal{Q}_{u-1}$ – the set of elements which are covered by an element of $\mathcal{Q}_u$. Then, for all $\mathcal{Q} \in \mathcal{A}_n$, $\frac{n+2}{2} \leqslant u \leqslant n$,

$$\mathcal{Q}' \in \mathcal{A}_n, \tag{7.4}$$

$$\Sigma(\mathcal{Q}') \geqslant \Sigma(\mathcal{Q}). \tag{7.5}$$

*Proof of* (7.4) (By contradiction) Suppose $\mathcal{Q}' \notin \mathcal{A}_n$. Then there exists $Q \in \mathcal{Q}_{u-1}$ such that $Q' \subseteq Q$ for some $Q' \in \mathcal{Q} \setminus \mathcal{Q}_u$. However, this implies that there exists $Q'' \in \mathcal{Q}_u$ such that $Q \subset Q''$ by construction of $\mathcal{Q}_{u-1}$, and hence that $Q' \subset Q''$ and $\mathcal{Q} \notin \mathcal{A}_n$. □

*Proof of* (7.5) We count $N$, the number of pairs $(Q, Q')$ such that $Q \in \mathcal{Q}_u$, $Q' \in \mathcal{Q}_{u-1}$ and $Q' \lessdot Q$, in two different ways.

- For a particular $Q \in \mathcal{Q}_u$ there are exactly $u$ such subsets $Q'$ (obtained by omitting one of the $u$ elements of $Q$).

- For a particular $Q' \in \mathcal{Q}_{u-1}$ there are $n-(u-1) = n-u+1$ possible subsets $Q$ which cover $Q'$, since $|Q'| = u-1$. However, not all of these are necessarily in $\mathcal{Q}_u$.

Therefore, we have

$$u|\mathcal{Q}_u| = N \leqslant (n-u+1)|\mathcal{Q}_{u-1}|. \tag{7.6}$$

Hence

$$\frac{|\mathcal{Q}_{u-1}|}{|\mathcal{Q}_u|} \geqslant \frac{u}{n-u+1} \geqslant \frac{u}{u-1} \quad \text{since } u \geqslant \frac{n+2}{2} \text{ implies } n-u+1 \leqslant u-1,$$

and therefore

$$(u-1)|\mathcal{Q}_{u-1}| \geqslant u|\mathcal{Q}_u|. \tag{7.7}$$

Now, by definition,

$$\Sigma(\mathcal{Q}') = \Sigma(\mathcal{Q}) - u|\mathcal{Q}_u| + (u-1)|\mathcal{Q}_{u-1}|,$$

and hence, using (7.6) and (7.7), we have

$$\Sigma(\mathcal{Q}') \geqslant \Sigma(\mathcal{Q}) \text{ with equality when } u = \frac{n+2}{2} \text{ and } u|\mathcal{Q}_u| = (n-u+1)|\mathcal{Q}_{u-1}|. \tag{7.8}$$

□

Similarly, let $l = \min_{Q \in \mathcal{Q}} |Q|$, and define

$$\begin{aligned} \mathcal{Q}_l &= \{Q \in \mathcal{Q} : |Q| = l\}, \\ \mathcal{Q}_{l+1} &= \{Q \subseteq [n] : \text{there exists } Q' \in \mathcal{Q}_l \text{ such that } Q' \lessdot Q\}, \\ \mathcal{Q}'' &= (\mathcal{Q} \setminus \mathcal{Q}_l) \cup \mathcal{Q}_{l+1}. \end{aligned}$$

Then, for all $\mathcal{Q} \in \mathcal{A}_n$, $0 \leqslant l \leqslant \frac{n}{2}$,

$$\mathcal{Q}'' \in \mathcal{A}_n, \tag{7.9}$$

$$\Sigma(\mathcal{Q}'') \geqslant \Sigma(\mathcal{Q}). \tag{7.10}$$

*Proof of* (7.9) (By contradiction) Suppose $\mathcal{Q}'' \notin \mathcal{A}_n$. Then there exists $Q \in \mathcal{Q}_{l+1}$ such that $Q \subseteq Q'$ for some $Q' \in \mathcal{Q} \setminus \mathcal{Q}_l$. However, this implies that there exists $Q'' \in \mathcal{Q}_l$ such that $Q'' \subset Q$ by construction of $\mathcal{Q}_{l+1}$, and that $Q'' \subset Q'$ and $\mathcal{Q} \notin \mathcal{A}_n$. □

*Proof of* (7.10) We count $N$, the number of pairs $(Q, Q')$ such that $Q \in \mathcal{Q}_l$, $Q' \in \mathcal{Q}_{l+1}$ and $Q \lessdot Q'$, in two different ways.

- For a particular $Q \in \mathcal{Q}_l$ there are exactly $n-l$ such subsets $Q'$ (obtained by adding one of the elements in $[n]$ not in $Q$).

- For a particular $Q' \in \mathcal{Q}_{l+1}$ there are $l+1$ possible subsets $Q$ which are covered by $Q'$, since $|Q'| = l+1$. However, not all these are necessarily in $\mathcal{Q}_l$.

Therefore, we have

$$(n-l)|\mathcal{Q}_l| = N \leqslant (l+1)|\mathcal{Q}_{l+1}| \tag{7.11}$$

Hence

$$\frac{|\mathcal{Q}_{l+1}|}{|\mathcal{Q}_l|} \geqslant \frac{n-l}{l+1} \geqslant \frac{l}{l+1} \quad \text{since } l \leqslant \frac{n}{2} \text{ implies } n-l \geqslant l,$$

and therefore

$$(l+1)|\mathcal{Q}_{l+1}| \geqslant l|\mathcal{Q}_l|. \tag{7.12}$$

By definition,

$$\Sigma(\mathcal{Q}'') = \Sigma(\mathcal{Q}) - l|\mathcal{Q}_l| + (l+1)|\mathcal{Q}_{l+1}|,$$

and hence by (7.12) we have

$$\Sigma(\mathcal{Q}'') \geqslant \Sigma(\mathcal{Q}) \quad \text{with equality when } l = \frac{n}{2} \text{ and } (n-l)|\mathcal{Q}_l| = (l+1)|\mathcal{Q}_{l+1}|. \tag{7.13}$$

□

Since, by assumption, $\Sigma(\mathcal{Q})$ is maximal, we have, by (7.8) and (7.13), $u \leqslant \frac{n+2}{2}$ and $l \geqslant \frac{n}{2}$. We now have three cases to consider, noting that, by definition, $l \leqslant u$.

- $n$ odd:

  then $u = l = \frac{n+1}{2}$ and $\mathcal{Q} = \mathcal{P}$.

- $n$ even, $l = u$:

  then either $u = l = \frac{n}{2}$ or $u = l = \frac{n+2}{2}$ and $\mathcal{Q} = \mathcal{P}$.

- $n$ even, $l < u$ (that is $l = \frac{n}{2}$, $u = \frac{n+2}{2}$ and hence $l = u-1$):

  then we derive a contradiction as follows. Since $\Sigma(\mathcal{Q})$ is assumed to be maximal we must have equality in (7.8). Therefore, $u|\mathcal{Q}_u| = (n-u+1)|\mathcal{Q}_{u-1}|$. In other words, for each $Q \in \mathcal{Q}_{u-1}$, every (covering) superset of $Q$ must be in $\mathcal{Q}_u$.

  Now choose some $Q' \in \mathcal{Q} \setminus \mathcal{Q}_u$ and $Q \in \mathcal{Q}_{u-1}$ such that $|Q' \cap Q|$ is maximal. We have $l = |Q'| = |Q| = u-1$ and $Q' \neq Q$. Hence there exists some $q' \in Q' \setminus Q$ and some $q \in Q \setminus Q'$ and, because $u|\mathcal{Q}_u| = (n-u+1)|\mathcal{Q}_{u-1}|$, $Q \cup \{q'\} \in \mathcal{Q}_u$, $Q'' = Q \cup \{q'\} \setminus \{q\} \in \mathcal{Q}_{u-1}$. Therefore, $|Q' \cap Q''| = |Q' \cap Q| + 1$ contradicting the maximality of $|Q' \cap Q|$.

■

The following alternative proof of Lemma 7.3.1 is based on a suggestion by Michael Saks and makes use of the LYM-inequality (page 28).

**Alternative Proof of Lemma 7.3.1** (Sketch) By Remark 2.2.1, we have, for any Sperner family $\mathcal{P}$ containing $n_k$ sets of cardinality $k$,

$$\sum_k \frac{n_k}{\binom{n}{k}} \leqslant 1.$$

Hence, we have

$$\sum_k \frac{kn_k}{k\binom{n}{k}} \leqslant 1.$$

Now

$$k\binom{n}{k} = (n-k+1)\binom{n}{k-1} \leqslant \lceil n/2\rceil \binom{n}{\lceil n/2\rceil},$$

with equality when

$$k = \begin{cases} \lceil n/2\rceil & \text{if } k \text{ is odd,} \\ n/2 \text{ or } (n+2)/2 & \text{if } n \text{ is even.} \end{cases}$$

Hence

$$\sum_k \frac{kn_k}{\lceil n/2\rceil \binom{n}{n/2}} \leqslant \sum_k \frac{kn_k}{k\binom{n}{k}} \leqslant 1,$$

which implies that

$$\sum_k kn_k = \Sigma(\mathcal{P}) \leqslant \lceil n/2\rceil \binom{n}{\lceil n/2\rceil}.$$

■

The remainder of the section is divided into two subsections. The first of these establishes a lower bound for $|\mathcal{A}_n|$, while the second establishes an upper bound for $|\mathcal{A}_n|$.

## 7.3.1 Lower bound for $|\mathcal{A}_n|$

In this section we prove several preparatory results which lead to Theorem 7.3.1. This theorem states the maximal cardinality of a conflict of interest policy containing a constraint of cardinality $r$. A corollary of this result supplies an improved lower bound for $|\mathcal{A}_n|$.

This section makes extensive use of the following propositions, which can be easily understood by considering Pascal's triangle (shown in Figure 7.3).

**Proposition 7.3.1** *For $1 \leqslant r \leqslant n$,*

$$\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}, \tag{7.14}$$

$$\binom{n}{r} = \binom{n}{n-r}, \tag{7.15}$$

$$\binom{n}{0} \leqslant \binom{n}{1} \leqslant \cdots \leqslant \binom{n}{\lfloor n/2 \rfloor} = \binom{n}{\lceil n/2 \rceil} \geqslant \cdots \geqslant \binom{n}{n-1} \geqslant \binom{n}{n}. \tag{7.16}$$

**Proof** By definition,

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}, \tag{7.17}$$

from which (7.14) and (7.15) follow immediately. Let $r \leqslant \lfloor n/2 \rfloor$. Then

$$\begin{aligned}\binom{n}{r} - \binom{n}{r-1} &= \frac{n!}{r!(n-r)!} - \frac{n!}{(r-1)!(n-r+1)!} \\ &= \frac{n!}{r!(n-r+1)!}(n-2r+1) \\ &\geqslant 0.\end{aligned}$$

The left hand side of (7.16) now follows; the right hand side follows from (7.15). ■

$$\begin{array}{c}
1 \\
1 \quad 1 \\
1 \quad 2 \quad 1 \\
1 \quad 3 \quad 3 \quad 1 \\
1 \quad 4 \quad 6 \quad 4 \quad 1 \\
1 \quad 5 \quad 10 \quad 10 \quad 5 \quad 1 \\
1 \quad 6 \quad 15 \quad 20 \quad 15 \quad 6 \quad 1 \\
1 \quad 7 \quad 21 \quad 35 \quad 35 \quad 21 \quad 7 \quad 1 \\
1 \quad 8 \quad 28 \quad 56 \quad 70 \quad 56 \quad 28 \quad 8 \quad 1 \\
1 \quad 9 \quad 36 \quad 84 \quad 126 \quad 126 \quad 84 \quad 36 \quad 9 \quad 1 \\
1 \quad 10 \quad 45 \quad 120 \quad 210 \quad 252 \quad 210 \quad 120 \quad 45 \quad 10 \quad 1 \\
1 \quad 11 \quad 55 \quad 165 \quad 330 \quad 462 \quad 462 \quad 330 \quad 165 \quad 55 \quad 11 \quad 1 \\
1 \quad 12 \quad 66 \quad 220 \quad 495 \quad 792 \quad 924 \quad 792 \quad 495 \quad 220 \quad 66 \quad 12 \quad 1
\end{array}$$

**Figure 7.3:** Pascal's triangle: $\binom{n}{r}$, $0 \leqslant r \leqslant n \leqslant 12$

**Lemma 7.3.2** *For $n \geqslant 1$, $0 \leqslant r \leqslant n$, let $\mathcal{P} \in \mathcal{A}_n$ have a smallest constraint of size $r$. Then*

$$|\mathcal{P}| \leqslant \begin{cases} \binom{n}{\lfloor n/2 \rfloor} - \binom{n-r}{\lfloor n/2 \rfloor - r} + 1 & \text{if } 0 \leqslant r \leqslant \lfloor n/2 \rfloor, \\ \binom{n}{r} & \text{if } \lfloor n/2 \rfloor \leqslant r \leqslant n, \end{cases}$$

*with equality when*

$$\mathcal{P} = \begin{cases} \{P'\} \cup \{P \subseteq [n] : |P| = \lfloor n/2 \rfloor,\ P' \not\subseteq P\} & \text{if } 0 \leqslant r \leqslant \lfloor n/2 \rfloor, \\ \{P \subseteq [n] : |P| = r\} & \text{if } \lfloor n/2 \rfloor \leqslant r \leqslant n, \end{cases} \tag{7.18}$$

*where $P' \subseteq [n]$ and $|P'| = r$.*

**Proof** Let $\mathcal{P} = \{P_1, \ldots, P_k\}$, where $r = |P_1| \leqslant |P_2| \leqslant \cdots \leqslant |P_k| = s$. The proof is very similar to the original proof of Sperner's Theorem. That is, we assume that $\mathcal{P}$ has maximal cardinality and show that this assumption imposes certain restrictions on the structure of $\mathcal{P}$.

We first consider the case $r \geqslant \lfloor n/2 \rfloor$. Define

$$\begin{aligned} \mathcal{P}^{(s)} &= \{P \in \mathcal{P} : |P| = s\}, \\ \mathcal{P}^{(s-1)} &= \left\{P' \subseteq [n] : |P'| = s-1,\ P' \lessdot P \text{ for some } P \in \mathcal{P}^{(s)}\right\}, \\ \mathcal{P}' &= \left(\mathcal{P} \setminus \mathcal{P}^{(s)}\right) \cup \mathcal{P}^{(s-1)}. \end{aligned}$$

Then $\mathcal{P}' \in \mathcal{A}_n$.

*Proof* Suppose that $\mathcal{P}' \notin \mathcal{A}_n$. Then there exist $P' \in \mathcal{P}^{(s-1)}$ and $P \in \mathcal{P} \setminus \mathcal{P}^{(s)}$ such that $P \subseteq P'$. However, by construction, there exists $P'' \in \mathcal{P}^{(s)}$ such that $P' \lessdot P''$. That is, $P \subseteq P''$ contradicting the fact that $\mathcal{P} \in \mathcal{A}_n$. □

Furthermore, if $s \geqslant \lceil n/2 \rceil$, then $|\mathcal{P}'| \geqslant |\mathcal{P}|$.

*Proof* Let $N$ denote the number of pairs $(P, P')$, where $P \in \mathcal{P}^{(s)}$ and $P' \in \mathcal{P}^{(s-1)}$. Then for each $P \in \mathcal{P}^{(s)}$ there are precisely $s$ such $P'$, and for each $P' \in \mathcal{P}^{(s-1)}$ there are $n - (s-1)$ possible $P$ (some of which may not be included in $\mathcal{P}^{(s)}$). Hence

$$N = s\left|\mathcal{P}^{(s)}\right| \leqslant (n-s+1)\left|\mathcal{P}^{(s-1)}\right|.$$

Therefore, if $s \geqslant \lceil n/2 \rceil$,

$$\frac{\left|\mathcal{P}^{(s-1)}\right|}{\left|\mathcal{P}^{(s)}\right|} \geqslant \frac{s}{n-s+1} \geqslant 1,$$

with equality if, and only if, $s$ is odd, $s = \lceil n/2 \rceil$ and every set obtained by adding an element to a member of $\mathcal{P}^{(s-1)}$ is in $\mathcal{P}^{(s)}$. □

Hence, $|\mathcal{P}|$ is maximal when all subsets have size $r$.

We next consider the case $r < \lfloor n/2 \rfloor$. Let $q$ be the minimal cardinality of an element of $\mathcal{P} \setminus \{P_1\}$. Define

$$\mathcal{P}^{(q)} = \{P \in \mathcal{P} : |P| = q\} \tag{7.19}$$

$$\mathcal{P}^{(q+1)} = \left\{P' \subseteq [n] : |P'| = q+1, P \lessdot P',\ \text{for some } P \in \mathcal{P}^{(q)},\ P_1 \not\subseteq P'\right\} \tag{7.20}$$

$$\mathcal{P}' = \left(\mathcal{P} \setminus \mathcal{P}^{(q)}\right) \cup \mathcal{P}^{(q+1)}. \tag{7.21}$$

Then, as before, $\mathcal{P}' \in \mathcal{A}_n$.

We now consider $N$, the number of pairs $(P, P')$, where $P \in \mathcal{P}^{(q)}$ and $P' \in \mathcal{P}^{(q+1)}$. Let $j$ be the number of sets in $\mathcal{P}^{(q)}$ that *intersect maximally* with $P_1$. (That is, for each such $P$, $|P_1 \cap P| = r - 1$.) Then, given such a set $P$, there are $n - q - 1$ possible choices for $P'$; otherwise there are $n - q$ choices for $P'$. Conversely, for each $P'$, there are $q + 1$ possible choices for $P$, although $P$ is not necessarily in $\mathcal{P}^{(q)}$. Hence

$$(n-q-1)j + (n-q)\left(\left|\mathcal{P}^{(q)}\right| - j\right) = N \leqslant (q+1)\left|\mathcal{P}^{(q+1)}\right|.$$

That is,

$$-j + (n-q)\left|\mathcal{P}^{(q)}\right| \leqslant (q+1)\left|\mathcal{P}^{(q+1)}\right|.$$

Hence

$$\frac{\left|\mathcal{P}^{(q+1)}\right|}{\left|\mathcal{P}^{(q)}\right|} \geqslant \frac{n-q}{q+1} - \frac{j}{\left|\mathcal{P}^{(q)}\right|(q+1)} \geqslant 1, \tag{7.22}$$

provided $\left|\mathcal{P}^{(q)}\right|(n-2q-1) \geqslant j$, which holds when $q < \lfloor n/2 \rfloor$ since $\left|\mathcal{P}^{(q)}\right| \geqslant j$. We have equality in (7.22) if, and only if, $n$ is even, $q = \lfloor n/2 \rfloor - 1$ and $\left|\mathcal{P}^{(q)}\right| = j$. (Note that $\left|\mathcal{P}^{(r+1)}\right| > \left|\mathcal{P}^{(r)}\right|$ unless $n$ is even and $r = \lfloor n/2 \rfloor - 1$.) Hence, excluding the exceptional case, we have $|\mathcal{P}'| > |\mathcal{P}|$. Since $\mathcal{P}$ is assumed to have maximal cardinality and discounting the exceptional case, we deduce that $q \geqslant \lfloor n/2 \rfloor$.

We know from the argument for the case $r \geqslant \lfloor n/2 \rfloor$ that $\mathcal{P}$ is maximal implies that $s \leqslant \lceil n/2 \rceil$. If $n$ is odd, either $q = s = \lfloor n/2 \rfloor$ or $q = s = \lceil n/2 \rceil$ or $q = s - 1 = \lfloor n/2 \rfloor$. In the second case, there are more sets of cardinality $\lceil n/2 \rceil$ that contain $P_1$ than there are of cardinality $\lfloor n/2 \rfloor$. (In particular, there are $\binom{n-r}{\lceil n/2 \rceil - r}$ and $\binom{n-r}{\lfloor n/2 \rfloor - r}$ sets of cardinality $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$, respectively. It can be shown that the former is greater than the latter provided $r > 0$.) In the third case, a similar argument to that employed at the end of Lemma 7.3.1 leads to a contradiction. Hence $q = s = \lfloor n/2 \rfloor$.

If $n$ is even, either $q = s = n/2$ or $r = q = s - 1 = n/2 - 1$ or $r = q = s = n/2 - 1$ and every subset of cardinality $q$ intersects maximally with $P_1$. In the second case, a similar argument to that employed at the end of Lemma 7.3.1 leads to a contradiction. In the third case, which corresponds to the exceptional case identified above, there are $\binom{n}{n/2-1}$ members in $\mathcal{P}$. However, a policy $\mathcal{P}'$, in which there is a single member of cardinality $n/2 - 1$, the remainder having cardinality $n/2$, has cardinality

$$\binom{n}{n/2} - (n/2 + 1) + 1 = \binom{n}{n/2} - n/2.$$

It can be shown that $\binom{n}{n/2} - n/2 > \binom{n}{n/2-1}$ for $n > 4$. Hence, the exceptional cases occur when $n = 2$, $r = 0$ and $n = 4$, $r = 1$.[1] ■

[1]For example, maximal policies when $n = 4$ and $r = 1$ are $\{\{1\}, \{2\}, \{3\}, \{4\}\}$ and $\{\{1\}, \{2,3\}, \{2,4\}, \{3,4\}\}$.

**Lemma 7.3.3** *For $n \geqslant 1$, $0 \leqslant r \leqslant n$, let $\mathcal{P} \in \mathcal{A}_n$ have a largest constraint of size $r$. Then,*

$$|\mathcal{P}| \leqslant \begin{cases} \binom{n}{r} & \text{if } 0 \leqslant r \leqslant \lceil n/2 \rceil, \\ \binom{n}{\lceil n/2 \rceil} - \binom{r}{\lceil n/2 \rceil} + 1 & \text{if } \lceil n/2 \rceil \leqslant r \leqslant n, \end{cases}$$

*with equality when*

$$\mathcal{P} = \begin{cases} \{P \subseteq [n] : |P| = r\} & \text{if } 0 \leqslant r \leqslant \lceil n/2 \rceil, \\ \{P'\} \cup \{P \subseteq [n] : |P| = \lceil n/2 \rceil, P \not\subseteq P'\} & \text{if } \lceil n/2 \rceil \leqslant r \leqslant n, \end{cases} \tag{7.23}$$

*where $P' \subseteq [n]$ and $|P'| = r$.*

**Proof** (Sketch) We note that the mapping $\psi : 2^{[n]} \to 2^{[n]}$, where $\psi(X) = [n] \setminus X$ extends in the natural way to a mapping $\Psi : \mathcal{A}_n \to \mathcal{A}_n$. Furthermore, if $\mathcal{P}$ has a largest constraint $P'$ of size $r$, then $\Psi(\mathcal{P})$ has a smallest constraint $\psi(P') = [n] \setminus P'$ of cardinality $n - r$. Hence, by Lemma 7.3.2, $\Psi(\mathcal{P})$ is maximal if every element (other than $P'$) has cardinality $\lfloor n/2 \rfloor$. That is, $\mathcal{P}$ is maximal if every element (other than $P'$) has cardinality $\lceil n/2 \rceil$. Finally, we note that there are $\binom{r}{\lceil n/2 \rceil}$ subsets of cardinality $\lceil n/2 \rceil$ that are contained in a subset of cardinality $r$. The result now follows. ■

**Definition 7.3.2** *Let $|P'| = r$ and*

$$\mathcal{P} = \begin{cases} \{P'\} \cup \{P \subseteq [n] : |P| = \lfloor n/2 \rfloor,\ P' \not\subseteq P\} & \text{if } 0 \leqslant r \leqslant \lfloor n/2 \rfloor, \\ \{P'\} \cup \{P \subseteq [n] : |P| = \lceil n/2 \rceil, P \not\subseteq P'\} & \text{otherwise.} \end{cases} \tag{7.24}$$

*Then we say $\mathcal{P}$ is a* default maximal $r$-policy. *The cardinality of $\mathcal{P}$ is denoted by $\left[{n \atop r}\right]$. For $0 \leqslant r \leqslant n$, we denote the set of default maximal $r$-policies by $\mathcal{A}_{n,r}$.*

Figure 7.4 shows the values of $\left[{n \atop r}\right]$, $0 \leqslant r \leqslant n \leqslant 12$.

```
                              1
                           1     1
                        1     2     1
                     1     3     3     1
                  1     4     6     4     1
               1     7    10    10     7     1
            1    11    17    20    17    11     1
         1    21    31    35    35    31    21     1
      1    36    56    66    70    66    56    36     1
   1    71   106   121   126   126   121   106    71     1
1   127   197   232   247   252   247   232   197   127     1
1   253   379   435   456   462   462   456   435   379   253     1
1   463   715   841   897   918   924   918   897   841   715   463     1
```

**Figure 7.4:** $\left[{n \atop r}\right]$ for $0 \leqslant r \leqslant n \leqslant 12$

**Lemma 7.3.4** *Let $n \geqslant 1$, $0 \leqslant r \leqslant q \leqslant n$, $\mathcal{P} \in \mathcal{A}_{n,r}$, $\mathcal{Q} \in \mathcal{A}_{n,q}$ and $\mathcal{P}' \subseteq \mathcal{P}$, where $r \neq \lfloor n/2 \rfloor$ or $\lceil n/2 \rceil$. Then, if $\mathcal{P}'$ contains the set of cardinality $r$ in $\mathcal{P}$, $\mathcal{P}' \nsubseteq \mathcal{Q}$.*

Despite its apparent complexity, this is not a profound result. Informally the lemma states that, for a fixed $r$, if a subset $\mathcal{P}'$ of a default maximal conflict of interest policy $\mathcal{P}$ contains the constraint of size $r$ in $\mathcal{P}$, then $\mathcal{P}'$ cannot be a subset of any other default maximal policy. For example, $\mathcal{A}_{4,1} = \{\{\{1\},\{2,3\},\{2,4\},\{3,4\}\},\ldots,\{\{4\},\{1,2\},\{1,3\},\{2,3\}\}\}$. Furthermore, if we choose a subset $\mathcal{P}'$ of $\{\{1\},\{2,3\},\{2,4\},\{3,4\}\}$ such that $\{1\} \in \mathcal{P}'$, then $\mathcal{P}'$ cannot be a subset of another element of $\mathcal{A}_{4,1}$ (since such a policy does not contain $\{1\}$), nor can it be a subset of any member of $\mathcal{A}_{4,k}$ for $k \neq 1$ (since such a policy does not contain a set of cardinality 1). Note also that $|\mathcal{A}_{4,1}| = \binom{4}{1}$.

**Proof of Lemma 7.3.4** Let $P' \in \mathcal{P}$, where $|P| = r$. There are two cases to consider.

- If $r = q$ then for all $\mathcal{Q} \neq \mathcal{P} \in \mathcal{A}_{n,r}$, $P' \notin \mathcal{Q}$, and hence $\mathcal{P}' \nsubseteq \mathcal{Q}$.
- If $r \neq q$ then for all $\mathcal{Q} \in \mathcal{A}_{n,q}$, $\mathcal{Q}$ contains no subset of size $r$ and hence $\mathcal{P}' \nsubseteq \mathcal{Q}$.

■

We finally obtain a lower bound for $|\mathcal{A}_n|$ by counting all subsets of $\mathcal{A}_{n,r}$, where $r$ ranges from 0 to $n$.

**Corollary 7.3.1** *For $n \geqslant 3$, $0 \leqslant r \leqslant n$, define*

$$\begin{vmatrix} n \\ r \end{vmatrix} = \begin{cases} 2 & \text{if } r = \lfloor n/2 \rfloor \text{ or } \lceil n/2 \rceil, \\ \binom{n}{r} & \text{otherwise.} \end{cases}$$

*Then*

$$|\mathcal{A}_n| \geqslant \sum_{r=0}^{n} \begin{vmatrix} n \\ r \end{vmatrix} 2^{\left[{n \atop r}\right]-1} > 2^{\nu}, \tag{7.25}$$

*where $\nu = \binom{n}{\lfloor n/2 \rfloor}$.*

**Proof** The result is proved by counting subsets of members of $\mathcal{A}_{n,r}$. We make the following observations.

- For $n \geqslant 1$,

$$|\mathcal{A}_{n,r}| = \begin{cases} 1 & \text{if } r = \lfloor n/2 \rfloor \text{ or } r = \lceil n/2 \rceil, \\ \binom{n}{r} & \text{otherwise.} \end{cases}$$

- All $2^{\binom{n}{\lfloor n/2 \rfloor}}$ subsets of $\mathcal{A}_{n,\lfloor n/2 \rfloor}$ and $\mathcal{A}_{n,\lceil n/2 \rceil}$ can be included (there is no distinguished constraint in terms of size). (Hence $\left|{n \atop \lfloor n/2 \rfloor}\right| = \left|{n \atop \lceil n/2 \rceil}\right| = 2$ in order to compensate for the $-1$ in the exponent in (7.25). Technically the empty set is counted twice when $n$ is odd, but for $n \geqslant 3$ this has no effect on the inequalities.)

- For all other values of $r$ and all $\mathcal{P} \in \mathcal{A}_{n,r}$, every subset of $\mathcal{P}$ that contains the constraint of size $r$ belongs to $\mathcal{A}_n$ and each is counted only once, by Lemma 7.3.4. There are $\binom{n}{r}$ such $\mathcal{P}$, and for each $\mathcal{P} \in \mathcal{A}_{n,r}$ there are $2^{|\mathcal{P}|-1}$ subsets which include the constraint of size $r$.
- We have $2^\nu = \left|{n \atop \lfloor n/2 \rfloor}\right| 2^{\left[{n \atop \lfloor n/2 \rfloor}\right]-1}$ which establishes the right-hand inequality.

The result now follows. ∎

We can summarize the results of this section in the following theorem which we state without proof.

**Theorem 7.3.1** *For all policies $\mathcal{P} \in \mathcal{A}_n$ containing a constraint $P'$ of cardinality $r$, $|\mathcal{P}| \leqslant \begin{bmatrix} n \\ r \end{bmatrix}$, with equality when*

$$\mathcal{P} = \begin{cases} \{P'\} \cup \{P \subseteq [n] : |P| = \lfloor n/2 \rfloor,\ P \not\subseteq P'\} & 0 \leqslant r \leqslant \lfloor n/2 \rfloor, \\ \{P'\} \cup \{P \subseteq [n] : |P| = \lceil n/2 \rceil,\ P' \not\subseteq P\} & \lceil n/2 \rceil \leqslant r \leqslant n. \end{cases}$$

We conclude this section with an identity involving the coefficients $\begin{bmatrix} n \\ r \end{bmatrix}$; it has a similar flavour to (7.14).

**Proposition 7.3.2** *For $n$ odd, $n \geqslant 1$, $0 \leqslant r \leqslant \lfloor n/2 \rfloor$,*

$$\begin{bmatrix} n+1 \\ r+1 \end{bmatrix} = \binom{n}{\lfloor n/2 \rfloor} + \begin{bmatrix} n \\ r \end{bmatrix}.$$

**Proof** We have

$$\begin{aligned} \begin{bmatrix} n+1 \\ r+1 \end{bmatrix} &= \binom{n+1}{\lceil (n+1)/2 \rceil} - \binom{n-r}{\lceil (n+1)/2 \rceil} + 1 \\ &= \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lceil n/2 \rceil} - \binom{n-r}{\lceil n/2 \rceil} + 1 \quad \text{(since } n \text{ is odd)} \\ &= \binom{n}{\lfloor n/2 \rfloor} + \begin{bmatrix} n \\ r \end{bmatrix}. \end{aligned}$$

∎

Particular cases of Proposition 7.3.2 can be seen by considering the appropriate rows of Figure 7.4. Note that it is far harder to establish a corresponding relation for $n$ even because, in this case, $\lfloor (n+1)/2 \rfloor = \lfloor n/2 \rfloor$ and $\lceil (n+1)/2 \rceil \neq \lceil n/2 \rceil$.

## 7.3.2 Upper bound for $|\mathcal{A}_n|$

We first define the novel concept of a *bi-symmetric chain partition.* The proof method of Theorem 2.2.3 and bi-symmetric chain partitions are used in the proof of Theorem 7.3.2, one of the main results in this chapter. An example of a bi-symmetric chain partition is shown in Table 7.5.

**Definition 7.3.3** *Let $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ be some symmetric chain partition of $2^{[n]}$. Then a* bi-symmetric chain partition *of $2^{[n+1]}$ is defined to be*

$$\{\mathcal{C}_1, \ldots, \mathcal{C}_k\} \cup \{\mathcal{C}_1 \cup \{n+1\}, \ldots, \mathcal{C}_k \cup \{n+1\}\},$$

*where $\mathcal{C} \cup \{n+1\}$ is the chain formed by taking the union of each element in $\mathcal{C}$ with $\{n+1\}$. The bi-symmetric chain partition of $2^{[n+1]}$ derived from $SCP_n$ is denoted by $BCP_n$.*

| | | | |
|---|---|---|---|
| $\mathcal{C}_1$ | $\emptyset \subset \{1\} \subset \{1,2\} \subset \{1,2,3\} \subset \{1,2,3,4\}$ | $\mathcal{D}_1$ | $\{5\} \subset \{1,5\} \subset \{1,2,5\} \subset \{1,2,3,5\} \subset \{1,2,3,4,5\}$ |
| $\mathcal{C}_2$ | $\{2\} \subset \{2,3\} \subset \{2,3,4\}$ | $\mathcal{D}_2$ | $\{2,5\} \subset \{2,3,5\} \subset \{2,3,4,5\}$ |
| $\mathcal{C}_3$ | $\{3\} \subset \{1,3\} \subset \{1,3,4\}$ | $\mathcal{D}_3$ | $\{3,5\} \subset \{1,3,5\} \subset \{1,3,4,5\}$ |
| $\mathcal{C}_4$ | $\{4\} \subset \{1,4\} \subset \{1,2,4\}$ | $\mathcal{D}_4$ | $\{4,5\} \subset \{1,4,5\} \subset \{1,2,4,5\}$ |
| $\mathcal{C}_5$ | $\{2,4\}$ | $\mathcal{D}_5$ | $\{2,4,5\}$ |
| $\mathcal{C}_6$ | $\{3,4\}$ | $\mathcal{D}_6$ | $\{3,4,5\}$ |

**Figure 7.5:** $BCP_5$

**Theorem 7.3.2** *For all $n \geqslant 3$,*

$$|\mathcal{A}_{n+1}| < 6^{\binom{n}{\lfloor n/2 \rfloor}} < 3^{\binom{n+1}{\lfloor (n+1)/2 \rfloor}}.$$

*Moreover,*

$$\lim_{n\to\infty} \frac{6^{\binom{n}{\lfloor n/2 \rfloor}}}{3^{\binom{n+1}{\lfloor (n+1)/2 \rfloor}}} = 0.$$

**Proof** To prove the right-hand inequality we make some preliminary observations.

- Using (7.14), we have for all $n > 1$,

$$\binom{n+1}{\lfloor (n+1)/2 \rfloor} = \binom{n}{\lceil n/2 \rceil} + \binom{n}{\lceil n/2 \rceil - 1}, \tag{7.26}$$

  and hence

$$1 < \binom{n}{\lfloor n/2 \rfloor} < \binom{n+1}{\lfloor (n+1)/2 \rfloor}. \tag{7.27}$$

- If $n$ is odd, then

$$\binom{n}{\lceil n/2 \rceil} = \binom{n}{\lfloor n/2 \rfloor} = \binom{n}{\lceil n/2 \rceil - 1}. \tag{7.28}$$

- For all $n > 1$,

$$\binom{2n}{n-1} = \frac{(2n)!}{(n-1)!(n+1)!} = \frac{(2n)!n}{(n!)(n!)(n+1)} = \frac{n}{n+1}\binom{2n}{n}. \tag{7.29}$$

- Finally, for all $n > 1$,

$$\frac{2}{3^{\left(\frac{n}{n+1}\right)}} < 1. \tag{7.30}$$

Hence, if $n$ is odd we have

$$\begin{aligned}
\frac{6^{\binom{n}{\lfloor n/2 \rfloor}}}{3^{\binom{n+1}{\lfloor (n+1)/2 \rfloor}}} &= \frac{2^{\binom{n}{\lfloor n/2 \rfloor}} 3^{\binom{n}{\lfloor n/2 \rfloor}}}{3^{\left(\binom{n}{\lceil n/2 \rceil} + \binom{n}{\lceil n/2 \rceil - 1}\right)}} && \text{by (7.26)} \\
&= \frac{2^{\binom{n}{\lfloor n/2 \rfloor}} 3^{\binom{n}{\lfloor n/2 \rfloor}}}{3^{\left(\binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor}\right)}} && \text{by (7.28)} \\
&= \frac{2^{\binom{n}{\lfloor n/2 \rfloor}} 3^{\binom{n}{\lfloor n/2 \rfloor}}}{3^{\binom{n}{\lfloor n/2 \rfloor}} 3^{\binom{n}{\lfloor n/2 \rfloor}}} \\
&= \left(\frac{2}{3}\right)^{\binom{n}{\lfloor n/2 \rfloor}} \\
&\to 0 \quad \text{as } n \to \infty && \text{by (7.27)}
\end{aligned}$$

Now suppose $n$ is even and let $n = 2m$, $m > 1$. Consider

$$\begin{aligned}
\frac{6^{\binom{n}{\lfloor n/2 \rfloor}}}{3^{\binom{n+1}{\lfloor (n+1)/2 \rfloor}}} &= \frac{6^{\binom{n}{\lfloor n/2 \rfloor}}}{3^{\binom{n}{\lceil n/2 \rceil}} 3^{\binom{n}{\lceil n/2 \rceil - 1}}} && \text{by (7.26)} \\
&= \frac{3^{\binom{2m}{m}} 2^{\binom{2m}{m}}}{3^{\binom{2m}{m}} 3^{\binom{2m}{m-1}}} \\
&= \frac{2^{\binom{2m}{m}}}{3^{\binom{2m}{m-1}}} \\
&= \frac{2^{\binom{2m}{m}}}{\left(3^{\left(\frac{m}{m+1}\right)}\right)^{\binom{2m}{m}}} && \text{by (7.29)} \\
&= \left(\frac{2}{3^{\left(\frac{m}{m+1}\right)}}\right)^{\binom{2m}{m}} \\
&\to 0 \quad \text{as } m \to \infty && \text{by (7.27) and (7.30)}
\end{aligned}$$

The result follows.

We now prove the left-hand inequality using a similar line of argument to that of Theorem 2.2.3. However, we consider a bi-symmetric chain partition $BCP_n$ rather than $SCP_n$. We will identify all chains in $BCP_n$ which are a copy of a chain in $SCP_{n-1}$ by $\mathcal{C}$, and all chains in $BCP_n$ which are of the form $\mathcal{C} \cup \{n\}$ by $\mathcal{D}$ (see Figure 7.5). If $C \in \mathcal{C}$ we denote the corresponding element in $\mathcal{D}$, $C \cup \{n\}$, by $D$. We will construct a filter $\mathcal{F}$ by choosing elements from each pair of corresponding chains $\mathcal{C}$ and $\mathcal{D}$.

Consider first a pair of chains of length at most two $C_0 \lessdot C_1$ and $D_0 \lessdot D_1$, say. By making our choices from $\mathcal{C}$ first, we can see there are at most 6 choices of elements from these two chains. In Table 7.2 we enumerate the choices available. A tick indicates the choice to include the element in $\mathcal{F}$, a cross indicates the choice to exclude the element from $\mathcal{F}$, and a hyphen indicates there is no choice involved because of the (subset inclusion) dependencies within the pairs of chains.

| $C_0$ | $C_1$ | $D_0$ | $D_1$ |
|---|---|---|---|
| ✓ | - | - | - |
| ✗ | ✓ | ✓ | - |
| ✗ | ✓ | ✗ | - |
| ✗ | ✗ | ✓ | - |
| ✗ | ✗ | ✗ | ✓ |
| ✗ | ✗ | ✗ | ✗ |

Table 7.2: The possible choices that can be made from a pair of chains, $C_0 \lessdot C_1$ and $D_0 \lessdot D_1$, in a bi-symmetric chain partition

Suppose now that we have to make a choice of elements from the chains

$$\mathcal{C}: \quad C_0 \lessdot \cdots \lessdot C_k \qquad \text{and} \qquad \mathcal{D}: \quad D_0 \lessdot \cdots \lessdot D_k$$

having already chosen the elements from the chains of shorter length, thus fixing some part of $\mathcal{F}$. By Theorem 2.2.4 we can find $C'_1, \dots, C'_{k-1}$ and $D'_1, \dots, D'_{k-1}$ belonging to shorter chains (than $\mathcal{C}$ and $\mathcal{D}$) such that

$$C_{i-1} \lessdot C'_i \lessdot C_{i+1} \quad \text{and} \quad D_{i-1} \lessdot D'_i \lessdot D_{i+1} \quad \text{for } 1 \leqslant i \leqslant k-1.$$

Define $l_c, u_c, l_d, u_d$ in an analogous way to $l$ and $u$ in Theorem 2.2.3. That is,

- $l_c$ is the largest integer such that $C'_{l_c} \notin \mathcal{F}$,
- $l_d$ is the largest integer such that $D'_{l_d} \notin \mathcal{F}$,
- $u_c$ is the smallest integer such that $C'_{u_c} \in \mathcal{F}$,
- $u_d$ is the smallest integer such that $D'_{u_d} \in \mathcal{F}$.

We first note that, as in Theorem 2.2.3, when $l_c > u_c$ or $l_d > u_d$, we will not be able to make any choices from $\mathcal{C}$ or $\mathcal{D}$, respectively.

Hence we can assume that $u_c - l_c = 1$ and $u_d - l_d = 1$ (that is, $C_{l_c} \lessdot C_{u_c}$ and $D_{l_d} \lessdot D_{u_d}$). (We would like to remind the reader at this point that $<$ will also be used to denote $\subset$. In this context, it is worth bearing in mind that $C_i$ and $D_i$ are sets and $l_c$, $l_d$, $u_c$ and $u_d$ are integers.)

There are the same 6 choices to extend $\mathcal{F}$ as given in Table 7.2 provided we can prove that, when we extend $\mathcal{F}$ by including zero or more of $C_{l_c}, C_{u_c}, D_{l_d}, D_{u_d}$, we do not contradict any decisions that have already been made with respect to the construction of $\mathcal{F}$. We consider the following two cases.

- Include $D_{l_d} \in \mathcal{D}$ in $\mathcal{F}$. We need to prove that $D_{l_d} \not< C'_i$, $1 \leqslant i \leqslant l_c$. (In other words, none of the elements $C'_1, \dots, C'_{l_c}$ already excluded from $\mathcal{F}$ will be included in $\mathcal{F}$ if $D_{l_d}$ is chosen to be included in $\mathcal{F}$. Clearly, if this holds then the inclusion of $D_{u_d}$ in $\mathcal{F}$ is legitimate since we can only include $D_{u_d}$ when $l_d < u_d$ and hence $D_{l_d} < D_{u_d} \not< C'_i$. Analogous remarks apply to the second case below.) As $n \in D$ for all $D \in \mathcal{D}$ and $n \notin C$ for all $C \in \mathcal{C}$, it is clearly the case that $D_{l_d} \not< C'_i$, $1 \leqslant i \leqslant l_c$.

- Include $C_{l_c} \in \mathcal{C}$ in $\mathcal{F}$. We need to prove that $C_{l_c} \not< D'_i$, $1 \leqslant i \leqslant l_d$ (for the same reasons cited in the case above). The proof proceeds as follows. We prove that

$$l_d \leqslant l_c \quad \text{and} \tag{7.31}$$

$$C_{l_c} \not< D'_{l_c}. \tag{7.32}$$

Then, by (7.31), $D'_{l_d} \leqslant D'_{l_c}$, and hence, by (7.32), $C_{l_c} \not< D'_{l_d}$.

*Proof of* (7.31) (By contradiction) Suppose that $l_c < l_d$. Then $C_{l_c} < C_{l_d}$ and $C'_{l_d} \in \mathcal{F}$ by definition of $l_c$. Therefore $D'_{l_d} \in \mathcal{F}$ since $C'_{l_d} \lessdot D'_{l_d}$, but, by definition, $D'_{l_d} \notin \mathcal{F}$. □

*Proof of* (7.32) (By contradiction) Suppose that $C_{l_c} < D'_{l_c}$. By construction, $|C_{l_c}| = |D'_{l_c}| - 1$, and $n \notin C_{l_c}$. Therefore, $|C_{l_c}| = |(D'_{l_c} \setminus \{n\})|$ and $C_{l_c} \leqslant (D'_{l_c} \setminus \{n\})$. Hence $C_{l_c} = (D'_{l_c} \setminus \{n\}) = C'_{l_c}$, which is a contradiction. □

We conclude by noting that no conflict can occur by including either $C_{u_c}$ or $D_{u_d}$ in $\mathcal{F}$. ■

As with Hansel's result, we can improve the upper bound for half the cases. Specifically, if we define

$$\nu(n) = \binom{n}{\lfloor n/2 \rfloor} \quad \text{and} \quad \mu(n) = \binom{n}{\lfloor n/2 \rfloor - 1},$$

we have the following result.

**Proposition 7.3.3** *For all $n \geqslant 3$,*

$$|\mathcal{A}_n| < 6^{\nu(n-1)} < 3^{\nu(n)},$$

*and for $n$ odd,*

$$|\mathcal{A}_n| < 3^{\nu(n-1)-\mu(n-1)} 6^{\mu(n-1)} = 2^{\mu(n-1)} 3^{\nu(n-1)}.$$

**Proof** $BCP_n$ is constructed using $SCP_{n-1}$ which necessarily contains $\nu(n-1) - \mu(n-1)$ chains of length 1. (This can be seen in Table 7.5, for example, in which there are $\binom{4}{2} - \binom{4}{1} = 2$ chains of length 1.)

For such pairs of chains, $C_0$ and $D_0$, we only have 3 choices of sets to include in $\mathcal{F}$. Namely, we can choose $D_0$ or $C_0$ (which necessarily includes $D_0$) or neither. ■

## 7.4 Postscript

The problem of determining $|\mathcal{A}_n|$ was first posed by Dedekind (1897) and is known to be very difficult. In particular, the value of $|\mathcal{A}_n|$ for $n \geqslant 9$ is not known. Table 7.3 shows the bounds for

$|\mathcal{A}_n|$ derived by Hansel and compares them with the corresponding values of $|\mathcal{A}_n|$ and $2^{2^n}$. The values of $|\mathcal{A}_n|$ are reproduced from Davey and Priestley (1990). Table 7.4 compares the values of $|\mathcal{A}_n|$ and the upper and lower bounds derived in this chapter with Hansel's bounds.

| $n$ | $2^{\nu(n)}$ | $\lvert\mathcal{A}_n\rvert$ | $3^{\nu(n)}$ | $2^{2^n}$ |
|---|---:|---:|---:|---:|
| 1 | 2 | 3 | 3 | 4 |
| 2 | 4 | 6 | 9 | 16 |
| 3 | 8 | 20 | 27 | 256 |
| 4 | 64 | 168 | 729 | 65536 |
| 5 | 1024 | 7581 | 59049 | 4294967296 |
| 6 | $1.048576 \times 10^{6}$ | $7.828354 \times 10^{6}$ | $3.486784 \times 10^{12}$ | $1.844674 \times 10^{18}$ |
| 7 | $3.435974 \times 10^{10}$ | $2.414682 \times 10^{12}$ | $5.003155 \times 10^{16}$ | $3.402824 \times 10^{38}$ |
| 8 | $1.180592 \times 10^{21}$ | $5.613044 \times 10^{22}$ | $2.503156 \times 10^{33}$ | $1.157921 \times 10^{77}$ |
| 9 | $8.507059 \times 10^{37}$ | ? | $1.310021 \times 10^{60}$ | $1.340781 \times 10^{154}$ |
| 10 | $7.237001 \times 10^{75}$ | ? | $1.716154 \times 10^{120}$ | $1.797693 \times 10^{308}$ |

Table 7.3: The upper and lower bounds of $|\mathcal{A}_n|$ due to Hansel

| $n$ | $2^{\nu(n)}$ | $\sum \lvert{n \atop r}\rvert 2^{[{n \atop r}]-1}$ | $\lvert\mathcal{A}_n\rvert$ | $6^{\nu(n-1)}$ | $3^{\nu(n)}$ |
|---|---:|---:|---:|---:|---:|
| 1 | 2 | 4 | 3 | 1 | 3 |
| 2 | 4 | 6 | 6 | 6 | 9 |
| 3 | 8 | 18 | 20 | 36 | 27 |
| 4 | 64 | 130 | 168 | 216 | 729 |
| 5 | 1024 | 2690 | 7581 | 46656 | 59049 |
| 6 | $1.048576 \times 10^{6}$ | $3.026946 \times 10^{6}$ | $7.828354 \times 10^{6}$ | $6.046618 \times 10^{7}$ | $3.486784 \times 10^{12}$ |
| 7 | $3.435974 \times 10^{10}$ | $1.138313 \times 10^{11}$ | $2.414682 \times 10^{12}$ | $3.656158 \times 10^{15}$ | $5.003155 \times 10^{16}$ |
| 8 | $1.180592 \times 10^{21}$ | $5.314680 \times 10^{21}$ | $5.613044 \times 10^{22}$ | $1.719071 \times 10^{27}$ | $2.503156 \times 10^{33}$ |
| 9 | $8.507059 \times 10^{37}$ | $3.934544 \times 10^{38}$ | ? | $2.955204 \times 10^{54}$ | $1.310021 \times 10^{60}$ |
| 10 | $7.237001 \times 10^{75}$ | $5.473068 \times 10^{76}$ | ? | $1.114442 \times 10^{98}$ | $1.716154 \times 10^{120}$ |

Table 7.4: A comparison of upper and lower bounds of $|\mathcal{A}_n|$

### 7.4.1 Asymptotic limits for $|\mathcal{A}_n|$

We now briefly describe work on the asymptotic behaviour of $|\mathcal{A}_n|$. It follows from Hansel's theorem that

$$\binom{n}{\lfloor n/2 \rfloor} \log 2 \leqslant \log |\mathcal{A}_n| \leqslant \binom{n}{\lfloor n/2 \rfloor} \log 3.$$

Kleitman and Makowsky (1975) established that

$$\lim_{n\to\infty} \log |\mathcal{A}_n| = \binom{n}{\lfloor n/2 \rfloor} \log 2$$

by applying probabilistic methods and Hansel's approach to the set of all symmetric chain partitions. An asymptotic formula for $|\mathcal{A}_n|$ was found by Korshunov using an extremely complicated argument, as can be seen in the following result.

**Theorem 7.4.1 (Korshunov 1980)**

$$\lim_{n\to\infty} |\mathcal{A}_n| = \begin{cases} 2^{\binom{n}{n/2}} e^{\binom{n}{(n-2)/2}\left(\frac{1}{2^{n/2}}+\frac{n^2}{2^{n+5}}-\frac{n}{2^{n+4}}\right)} & \textit{if } n \textit{ is even,} \\ 2^{\binom{n}{(n-1)/2}+1} e^{\binom{n}{(n-3)/2}\left(\frac{1}{2^{(n+3)/2}}-\frac{n^2}{2^{n+6}}-\frac{n}{2^{n+3}}\right)+\binom{n}{(n-1)/2}\left(\frac{1}{2^{(n+1)/2}}+\frac{n^2}{2^{n+4}}\right)} & \textit{if } n \textit{ is odd.} \end{cases}$$

This asymptotic limit provides a good approximation of $|\mathcal{A}_n|$ even for small values of $n$. We are currently investigating a generalization of the bi-symmetric chain partition that enables us to improve our upper bound. A preliminary discussion of our ideas in this area appears in Chapter 9.

## 7.5 Discussion

We have presented a general framework for the articulation of conflict of interest policies which includes negative authorization policies and separation of duty policies as special cases. We believe our approach offers a simple and complete characterization of such policies, and significantly extends the class of policies for role-based access control.

We have not restricted our attention to conflict of interest policies in which the constraints are (mutually exclusive) pairs, but noted in Section 7.2 that separation of duty policies are usually modelled in this way. The only exception we have found is RCL 2000, but the fragments of the language the authors offer as examples suggest that although the constraints in a policy may have cardinality greater than two, the policy is violated if any pair of elements from a constraint enters the environment.

With this in mind, we can rewrite an arbitrary conflict of interest policy $\mathcal{P} \in \mathcal{A}_n$ as a policy $\mathcal{P}' \in \mathcal{A}_n^2$, where $\mathcal{A}_n^2 = \{\mathcal{P} \in \mathcal{A}_n : \text{for all } P \in \mathcal{P},\ |P| \leqslant 2\}$. Specifically, let

$$\mathcal{P} = \{P_i : 1 \leqslant i \leqslant n\},$$

then

$$\mathcal{P}' = \left\{\left\{a_{i_j}, a_{i_k}\right\} : 1 \leqslant j < k \leqslant |P_i|, |P_i| > 1, 1 \leqslant i \leqslant n\right\} \cup \left\{P_i : |P_i| = 1, 1 \leqslant i \leqslant n\right\}.$$

In other words, for all $P_i \in \mathcal{P}$, if $|P_i| = 1$ then include $P_i$ in $\mathcal{P}'$; otherwise replace $P_i$ by all pairs of elements in $P_i$. Clearly $\mathcal{P}' \leqslant \mathcal{P}$ with equality when $|P_i| \leqslant 2$, for all $1 \leqslant i \leqslant n$, so $\mathcal{P}'$ is, in general, more restrictive than $\mathcal{P}$. Therefore, an arbitrary conflict of interest policy, $\mathcal{P} \in \mathcal{A}_n$, can be expressed as a conflict of interest policy, $\mathcal{P}' \in \mathcal{A}_n^2$, which is at least as strong as $\mathcal{P}$. It can easily be seen that

$$2^{\binom{n}{2}} < |\mathcal{A}_n^2| < 3^{\binom{n}{2}},$$

since there are $\binom{n}{2}$ distinct pairs, and any subset of the set of pairs is a valid policy. Furthermore, for all $\mathcal{P} \in \mathcal{A}_n^2$,

$$\Sigma(\mathcal{P}) \leqslant 2\binom{n}{2} = n(n-1).$$

Therefore, assuming that it takes constant time to determine whether $x \in E$ for any element $x \in X$, the complexity of checking whether adding $x$ to $E$ will violate a policy in $\mathcal{A}_n^2$ is $\mathcal{O}(n^2)$.

Lemma 7.3.1 shows that at worst we will require

$$\lceil n/2 \rceil \binom{n}{\lceil n/2 \rceil}$$

tests to determine whether adding $x \in X$ to $E$ will violate a policy $\mathcal{P} \in \mathcal{A}_n$. It is clear, therefore, that implementing conflict of interest policies using elements of $\mathcal{A}_n^2$ will in general be far more efficient than using unrestricted elements of $\mathcal{A}_n$.

Hence, if the usual assumptions are made about the definition of conflict of interest policies, the complexity of such policies can be readily described. However, we feel that the effort involved in investigating the general case has been worthwhile. In particular, it has led to the generalization of a lattice of antichains in Chapter 6, which in turn provides a rigorous foundation for the material in Chapter 8.

The most interesting contribution to access control policies is the identification of several novel applications of conflict of interest policies (and RCL 2000). In particular, the ability to impose a ceiling on user-role assignments provides a useful adjunct to the administration of user-role assignment and a simpler approach to modelling mandatory access control in role-based access control.

# Chapter 8

# The Secure Hierarchical Authorization Framework

In this chapter we present the secure hierarchical authorization (SHA) framework which is a template for the design of access control models that address some of the shortcomings of existing models. Specifically, models based on the SHA framework combine stronger security properties than those associated with the protection matrix model and the role-based access control model with a more flexible approach to security requirements than the Bell-LaPadula model.

The development of the model is motivated by two observations. Firstly, the security lattice $C \times 2^K$ in the Bell-LaPadula model is a sublattice of the antichains in the partial order $C \cup K$. (This simple result is proved as part of Proposition 8.2.1.) Secondly, most organizations have a *supervision hierarchy* based on superiority of position within the organization.

> *" ... a hierarchy of named positions ... each position has one or more roles ... these roles can be supervisory (administrative), or functional or both."*
>
> Moffett and Lupu (1999)

It seems reasonable, therefore, to assume that access to objects can be identified with positions in a supervision hierarchy. This is essentially a combination of the ideas of a user-role assignment relation in the role-based access control model and the security clearance function of the Bell-LaPadula model. Using the obvious parallel with security labels in the Bell-LaPadula model, we informally have the idea of a security policy in which access to an object is granted to a subject only if the position of the subject is greater than that of the object.

We assume that objects and subjects can be associated with a set of positions, thus giving our model greater flexibility than the Bell-LaPadula model. By regarding the supervision hierarchy as a partial order on a set of positions and using the results of Chapter 6, we show how to extend the ordering of the supervision hierarchy to an ordering on sets of positions in a completion of the supervision hierarchy.

We also take the view that any non-policy-specific access control model can be used as the reference monitor in the Bell-LaPadula model. Specifically, we could use either a protection matrix or the "machinery" of the role-based access control model. The choice would be determined by the application domain.

In the remainder of this section we describe the components of the SHA framework. We assume the existence of the following:

- A partially ordered set of *positions* $\langle E, \leqslant \rangle$; the Hasse diagram of $E$ defines the *position hierarchy* and corresponds to the supervision hierarchy mentioned above. ($E$ denotes enterprise and is used because $P$ will be used for the set of permissions.)

  By Theorem 6.1.1, there exist two completions of $E$ using antichains (that is, two complete lattices, which preserve the ordering of $E$), namely $\langle \mathcal{A}(E), \preccurlyeq \rangle$ and $\langle DM_{\mathcal{A}}(E), \preccurlyeq \rangle$.

- A *reference monitor* which may be based on the protection matrix model, the role-based access control model, or any other policy-neutral access control model. The interpretation of subjects will be determined by the reference monitor chosen. For example, in the case of a role-based access control reference monitor, subjects will be interpreted as sessions.

- A *seniority function* $\phi$ which associates every entity in the system with a level of seniority within the organizational framework. This is clearly analogous to the security clearance function $\lambda$ in the Bell-LaPadula model.

  The domain of $\phi$ will depend on the reference monitor. For example, using a role-based access control reference monitor, we would define $\phi$ for all objects, roles and users.

  If the domain of $\phi$ is a partially ordered set, then we require that $\phi$ be *monotonic*. That is, for all $x, y \in dom(\phi)$, $x \leqslant y$ implies $\phi(x) \preccurlyeq \phi(y)$.

  The range of $\phi$ is $\mathcal{A}(E)$. In other words, the seniority of an entity is a set of positions which are pairwise incomparable (with respect to the position hierarchy). Typically, this set will be a singleton and hence will coincide with the usage of $\lambda$ in the Bell-LaPadula model. However, the fact that $\mathcal{A}(E)$ is a complete lattice means that a more senior antichain of positions can be found for any antichain of positions, and hence that this definition of $\phi$ is both legitimate and provides more flexibility than the definition of $\lambda$. We note the following possibilities.

  – $\phi(o) = \emptyset$: that is, $o$ is not protected, as every subject will be senior to $o$. This corresponds intuitively to public documents, for example.

  – $\phi(s) = \emptyset$: that is, $s$ has no access privileges except to public documents. This could model, for example, a guest user.

- An *order-based policy* (or *seniority policy* or simply *policy*) $\Pi$ which is a set of (policy) statements (or security properties), $\{\pi_1, \ldots, \pi_n\}$, where each $\pi_i$ is an inequality relating the seniority of entities within the system. Informally, an order-based policy is one in which the specification of the policy can be stated in terms of the ordering on the position hierarchy. We now give some examples of policy statements:

  – $\pi_{ssp} \triangleq \phi(s) \succcurlyeq \phi(o)$ is analogous to the simple security property;

  – $\pi_* \triangleq \phi(s) \preccurlyeq \phi(o)$ is analogous to the *-property.

  The policy $\Pi_{\text{BLP}} = \{\pi_{ssp}, \pi_*\}$ is analogous to the information flow policy of the Bell-LaPadula model.

- A *policy monitor* which is used to enforce an *order-based policy.*

  Our model, therefore, like the Bell-LaPadula model, has a two-stage protocol for determining the system response to a request. That is, the request must satisfy the seniority policy and it must be granted by the reference monitor.

  For example, if we assume the existence of a role-based access control reference monitor, a request $\texttt{get}(s, p)$, where $s$ is a session and $p$ is a permission, is granted if there exists $r \in R(s)$ such that $\phi(r) \succcurlyeq \phi(p)$ and $(p, r) \in PA$. We note that we can assign different positions to the permissions $p_r = (o, \texttt{read})$ and $p_w = (o, \texttt{write})$, for example, which provides more flexibility than the Bell-LaPadula model.

The SHA framework itself is too general to have any specific application. It is therefore necessary to consider particular cases in which security properties are defined and a reference monitor chosen. A *secure hierarchical authorization model* is an instance of the SHA framework in which the seniority policy and the reference monitor have been chosen. A *secure hierarchical system* $\mathsf{S}$ is an instance of some secure hierarchical authorization model. A *state* $t$ is a description of $\mathsf{S}$ at a particular instant. The (possibly non-deterministic) behaviour of $\mathsf{S}$ is defined by the initial state of the system, a set of requests $\Gamma$, and other features which are determined by the reference monitor. An *evolution* $t^n$ of $\mathsf{S}$ is a sequence of states $t_0 \dots t_n$, where $t_0$ is the initial state of the system and $t_i$ is obtained from $t_{i-1}$ by the execution of a request in $\Gamma$.

For a given policy statement $\pi$, we must define what is meant for a state to *satisfy* $\pi$. This definition will depend on $\pi$, just as the conditions for a state (in the Bell-LaPadula model) to satisfy the simple security property and the *-property are different.

To summarize, the SHA framework has the following characteristic features:

(1) The secure hierarchical authorization framework is an extension of the Bell-LaPadula model.

(2) The strict order of security labels and the set of categories are replaced by a more general hierarchy reflecting organizational characteristics.

(3) The security clearance function is replaced by a more general seniority function.

(4) Policies may be any combination of order-based statements.

(5) Two stage checking of access requests.

These features suggest the following advantages over existing models.

- (2), (3) and (4) will provide greater flexibility than the Bell-LaPadula model.
- (1) and (5) will ensure stronger security properties than existing discretionary access control models.

In the remainder of this chapter we consider particular SHA models. In Section 8.1 we describe the SHRBAC model which is a role-based access control model. We define the security properties of the model and the notion of a secure state and secure system. We then define a set of commands and prove that a SHRBAC system which implements these commands and whose initial state is secure will only enter secure states. (This result is analogous to the Basic Security Theorem of Bell and LaPadula (1973b). Indeed, the structure of this chapter as a whole is strongly influenced by the work of Bell and LaPadula.) In Section 8.2, we use a protection matrix as the reference monitor and show that the Bell-LaPadula model and the typed access matrix model (Sandhu 1992c) are both SHA models. We also briefly describe how a SHA model could be used to implement a transition secure system (McLean 1994).

## 8.1 SHRBAC

Several papers have appeared in recent years that demonstrate how role-based access control can be used to simulate mandatory access control. The most comprehensive treatment (Osborn et al. 2000) requires the use of a "read" hierarchy and a "write" hierarchy. Both hierarchies are lattices, the latter being the dual of the former. Two distinct hierarchies are required because permissions are inherited upwards in a role hierarchy. Informally, the *-property implies that "write" permissions are inherited downwards. In the next chapter, we discuss some preliminary ideas for an alternative approach to this topic.

SHRBAC does not attempt to simulate the Bell-LaPadula model. SHRBAC is instead an attempt to broaden the applicability of multi-level secure systems using role-based concepts. At this stage, we do not distinguish between different modes of access. That is, we focus on providing a security property analogous to the simple security property.

The SHRBAC model necessarily has two components which we shall call *mandatory* and *discretionary*. The mandatory part defines the security properties (defined in terms of the position hierarchy), while the discretionary part defines those aspects of the model that would be configured by the end-users. They correspond to the security properties and protection matrix parts of the Bell-LaPadula model, respectively. In Section 8.1.1 we describe the discretionary components of SHRBAC: the role hierarchy $RH$ and the relations $UA$ and $PA$. In Section 8.1.2 we define the security properties associated with the SHRBAC model: $\pi_{ra}$, $\pi_{pu}$, $\pi_{rh}$, $\pi_{ua}$, $\pi'_{ua}$, $\pi_{pa}$ and $\pi'_{pa}$.

We will assume the use of the SARBAC administrative model described in Chapter 4. However, in this chapter we will use $\text{RHA}_1$ rather than $\text{RHA}_4$ as the basis for SARBAC. In particular, administrative scope is defined by the role hierarchy, not the `admin-authority` relation. We choose this approach in order to simplify the exposition of SHRBAC and the description of requests. A preliminary analysis suggests that the use of an extended hierarchy and the `admin-authority` relation will be straightforward. We discuss the issues involved at the end of this section.

### 8.1.1 Discretionary components of SHRBAC

**Role hierarchy** The role hierarchy is defined by the covering relation of the partial order on $R$. In other words, we include a relation $RH \subseteq R \times R$ such that $(r, r') \in RH$ if, and only if, $r \lessdot r'$ in $R$. Therefore, our approach to the role hierarchy is the same as that employed in the NIST role-based access control model, but differs from RBAC96.

**User- and permission-role assignment** The assignment relations $UA$ and $PA$ refine the requirements imposed by $\pi_{ua}$ and $\pi_{pa}$ (defined below). This is analogous to the function of the protection matrix in the Bell-LaPadula model, which refines the simple security property and the

*-property. We will illustrate this with an example in Section 8.1.3. We assume that for all $u \in U$, $R(u) \in \mathcal{A}(R)$, and, for all $p \in P$, $R(p) \in \mathcal{A}(R)$.

**Sessions** We assume a user $u$ initiates a session $s$ by activating a set of roles $R(s) \in \mathcal{A}(R)$, where $R(s) \subseteq \downarrow R(u)$. We represent a session as a triple $(i, u, \downarrow R(s))$, where $i$ is a (unique) session identifier. If $s$ requests permission $p$, the reference monitor finds each $r \in R(p)$ and tests whether $r \in \downarrow R(s)$. If there exists $r \in R(p) \cap \downarrow R(s)$ and mandatory requirements are also satisfied, then access to $p$ is granted. (The assumption here is that a session will typically make many access requests and hence it will be more efficient to compute $\downarrow R(s)$ once, rather than computing $\uparrow R(p)$ for each request to use permission $p$.)

### 8.1.2 Security properties

Given an administrative role $a$, a role $r$, a user $u$, and a permission $p$, we define seven security properties for SHRBAC.

**Role activation:** $\pi_{ra} \triangleq \phi(u) \succcurlyeq \phi(r)$**.** The role activation property imposes a restriction on the roles a user can activate. Namely, the user must be at least as senior as the role.

*UA* satisfies $\pi_{ra}$ if for all $(u, r) \in UA$, $\phi(u) \succcurlyeq \phi(r)$. We say a session $s$ satisfies $\pi_{ra}$ if for all $r \in R(s)$, $\phi(u) \succcurlyeq \phi(r)$, where $u$ is the user running the session $s$.

**Permission usage:** $\pi_{pu} \triangleq \phi(r) \succcurlyeq \phi(p)$**.** The permission usage property imposes a restriction on the permissions a role can use. Namely, the role must be at least as senior as the permission.

*PA* satisfies $\pi_{pu}$ if for all $(p, r) \in PA$, $\phi(r) \succcurlyeq \phi(p)$.

**Hierarchy consistency:** $\pi_{rh} \triangleq r \leqslant r'$ implies $\phi(r) \preccurlyeq \phi(r')$. The hierarchy consistency property requires that $\phi$ be monotonic.

*RH* satisfies $\pi_{rh}$ if for all $(r, r') \in RH$, $\phi(r) \preccurlyeq \phi(r')$.

**User assignment:** $\pi_{ua} \triangleq \phi(a) \succ \phi(u) \succcurlyeq \phi(r)$**.** The user assignment property imposes restrictions on user-role assignment. Namely, the user must be at least as senior as the role to which the user is to be assigned, and the role performing the assignment must be more senior than the user.

**Weak user assignment:** $\pi'_{ua} \triangleq \phi(a) \succ \phi(u),\ \phi(a) \succ \phi(r)$**.** The weak user assignment property requires that both the user and the role be less senior than the role performing the assignment. Note that if we assume that *RH* satisfies $\pi_{rh}$, then we can omit the condition $\phi(a) \succ \phi(r)$ from $\pi'_{ua}$. A similar remark applies to weak permission assignment below. Note also that the property $\pi_{ua}$ is interchangeable with the pair of properties $\{\pi_{ra}, \pi'_{ua}\}$, since if we restrict user assignments to only those roles which are no more senior than the user, then the role activation property is redundant.

**Permission assignment:** $\pi_{pa} \triangleq \phi(a) \succ \phi(r) \succcurlyeq \phi(p)$**.** The permission assignment property requires that the role must be at least as senior as the permission, and the role performing the assignment must be more senior than the role.

**Weak permission assignment:** $\pi'_{pa} \triangleq \phi(a) \succ \phi(r),\ \phi(a) \succ \phi(p)$. The weak permission assignment property requires that both the permission and the role be less senior than the role performing the assignment. (The property $\pi_{pa}$ is interchangeable with the pair of properties $\{\pi_{pu}, \pi'_{pa}\}$.)

**Remark 8.1.1** *Note that the "weak" properties can be checked more efficiently than their counterparts. For example, to check weak user assignment we only need to test whether $\phi(u)$ and $\phi(r)$ are in $\downarrow\phi(a)$. However, to check the user assignment property requires testing whether $\phi(u)$ is in $\downarrow\phi(a)$ and whether $\phi(r)$ is in $\downarrow\phi(u)$.*

**Definition 8.1.1** *A* state *$t = \langle RH, UA, PA, S, V\rangle$ is determined by the role hierarchy, the user-permission assignment relation, the permission role-assignment relation, the set of sessions and $V$, which models the permissions currently granted by the system to sessions.*

*A* state *$t_i$ denotes the tuple $\langle RH_i, UA_i, PA_i, S_i, V_i\rangle$. An* evolution *$t^n$ is defined to be a sequence of states $t_0 \ldots t_n$, where $t_i$ is the result of executing some request in state $t_{i-1}$.*

**Definition 8.1.2** *A* SHRBAC system *$\mathsf{S}(P, \Gamma, U, E, \phi, t_0)$ is determined by the set of permissions, the set of requests, the set of users, the position hierarchy, the seniority function and the initial state of the system.*

**Remark 8.1.2** *A request in SHRBAC causes a change in the state of the system. Therefore, Definition 8.1.2 implies that $U$, $O$ and $\phi$ are fixed. In the Bell-LaPadula model $U$ and $O$ are also fixed; "dormant" subjects and objects are activated and assigned a security level. In other words, tranquillity is a feature of the current version of SHRBAC. Modelling changes to the seniority function is beyond the scope of this thesis.*

**Definition 8.1.3** *A state, $\langle RH, UA, PA, S, V\rangle$, is* secure *if for all $(u,p) \in V$, $\phi(u) \succcurlyeq \phi(p)$. A state is* consistent *if $RH$ satisfies $\pi_{rh}$. A state satisfies $\pi_{ra}$ if $UA$ satisfies $\pi_{ra}$ and for all $s \in S$, $s$ satisfies $\pi_{ra}$. A state satisfies $\pi_{pu}$ if $PA$ satisfies $\pi_{pu}$.*

*A state evolution, $t^n = t_0 t_1 \ldots t_n$, is* secure *if $t_i$ is secure, $0 \leqslant i \leqslant n$. A SHRBAC system is* secure *if all state evolutions are secure. A state evolution is consistent if $t_i$ is consistent, $0 \leqslant i \leqslant n$. A SHRBAC system is consistent if all state evolutions are consistent. A state evolution satisfies $\pi_{ra}$ and $\pi_{pu}$ if $t_i$ satisfies $\pi_{ra}$ and $\pi_{pu}$, $0 \leqslant i \leqslant n$. A SHRBAC system satisfies $\pi_{ra}$ and $\pi_{pu}$ if all state evolutions satisfy $\pi_{ra}$ and $\pi_{pu}$.*

Clearly the definition of a secure state is analogous to a state satisfying the simple security property in the Bell-LaPadula model. However, in Definition 8.1.3, there is no condition analogous to a state satisfying the *-property in the Bell-LaPadula model.

### 8.1.3 Hierarchy operations in SHRBAC

In this section we consider the effect of hierarchy operations on both the role hierarchy and sessions. The impact of hierarchy operations in an operational role-based system is not widely discussed in the literature. Therefore, we feel that these operations should be fully understood before defining requests.

The roles in a session are assumed to be static and calculated when the session is initiated. We assume that the addition of edges and roles to the role hierarchy (which can only increase the permissions available to roles) can proceed without interrupting sessions. However, the deletion of edges and roles from the hierarchy may reduce the permissions available. In order to maintain the security properties of the system, it is therefore necessary to suspend or stop any sessions which include the roles affected by such a hierarchy operation and to re-calculate the set of roles available to the session before resuming it.

## Edge insertion

The insertion of an edge must preserve the partial order and must not introduce transitive edges in $RH$. Therefore, to insert an edge $(r_c, r_p)$, we must have $r_c \parallel r_p$. (If $r_c < r_p$, then either $r_c \lessdot r_p$ and $(r_c, r_p) \in RH$ or $(r_c, r_p)$ is a transitive edge and should not be added to $RH$; if $r_p > r_c$, then adding $(r_c, r_p)$ to $RH$ would introduce a cycle into the role hierarchy.)

We wish to maintain $RH$ as the covering relation of the partial order on $R$. This means that certain edge insertions will require the deletion of transitive edges formed by the insertion. (By transitive edge, we mean $(r, r') \in RH$ such that there exists $r'' \in R$ with $(r, r'') \in RH$ and $(r'', r') \in RH$.) Figure 8.1 illustrates this point: the transitive edges $(r, r'')$ and $(r', r''')$, which arise as a result of inserting the edge $(r', r'')$, are deleted.

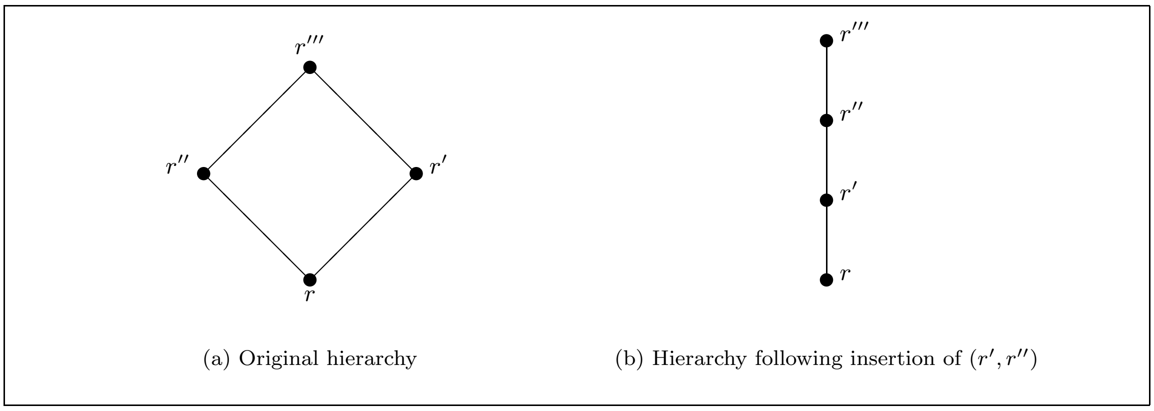


**Figure 8.1:** Edge insertion

## Edge deletion

The mechanics of edge deletion require more care. Edge deletion may reduce the permissions available to roles (through inheritance) and the permissions available to users and sessions. Hence, in an operational system, it seems reasonable to require that if an edge $(r, r')$ is to be deleted, then all sessions that contain $r'$ should be suspended or stopped.

Paradoxically, the deletion of an edge may also require the addition of new edges. For example, suppose $r \lessdot r' \lessdot r'' \lessdot r'''$ and we delete the edge $(r', r'')$. The following question arises: Should we now add the edges $(r, r'')$ and $(r', r''')$ to $RH$? We choose to support both possible answers, with "no" corresponding to *inheritance destruction* or *strong* edge deletion, and "yes" corresponding to *inheritance preservation* or (*weak*) edge deletion. In other words, weak edge deletion makes explicit any implied transitive edges that would be lost following an edge deletion, while strong edge deletion does not. (Note that we do not support a mixture of weak and strong edge deletion in which, say, $(r, r'')$ is added, but $(r', r''')$ is not.) Figure 8.2 illustrates the effect of weak and strong edge deletion of the edge $(r', r'')$. We anticipate that weak edge deletion would be far more common as it will preserve the inheritance of user- and permission-role assignments. Note that weak edge deletion is the inverse operation of edge insertion.

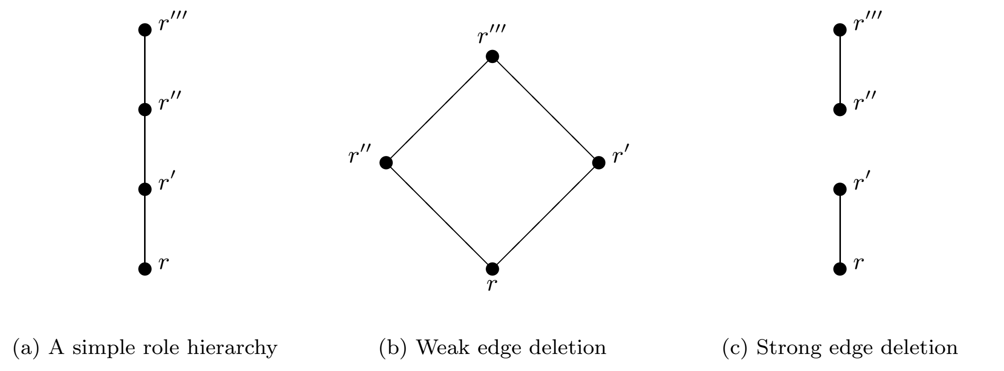


**Figure 8.2:** Edge deletion

### Role insertion

Role insertion can only increase the permissions available. We assume, therefore, that role insertion can be performed at any point, but current sessions will not reflect the addition of roles to the hierarchy. Role insertion generally requires the insertion of one or more edges, and therefore we may need to delete one or more transitive edges. For example, adding the role $r_1$ to the hierarchy in Figure 8.2a with parent $r'$ and child $r$ requires the deletion of the resulting transitive edge $(r, r')$.

### Role deletion

The deletion of a role, $r$, involves the deletion of all edges of the form $(r, r')$ and $(r'', r)$ in $RH$. Therefore, we support two strategies: *weak* role deletion and *strong* role deletion. Weak role deletion inserts transitive edges that are implied by $RH$ but would be lost by deleting $r$, while strong role deletion does not. Figure 8.3 illustrates the effect of deleting role $r'$ in Figure 8.2a.

We need to consider the effect of role deletion on the user- and permission-role assignments. We adopt a similar strategy to that used in ARBAC97. Namely, if we delete the role $r$ we re-assign all users from $r$ to each role in $\Delta r$, and all permissions from $r$ to each role in $\nabla r$. We do not consider the effect of role deletion on `ua-constraints` and `pa-constraints`.

### Summary and example

Role-based administration in SHRBAC is very simple and is determined by the role hierarchy and the implied administrative scope in the hierarchy. In Table 8.1 we summarize the important features of the SHRBAC model and their counterparts in the RBAC96/ARBAC97 models.

Figure 8.4 shows the components of a SHRBAC system based on the RBAC96/ARBAC97 example in Figure 3.2. Note the use of $UA$ to refine the roles that `bill` can activate. By virtue

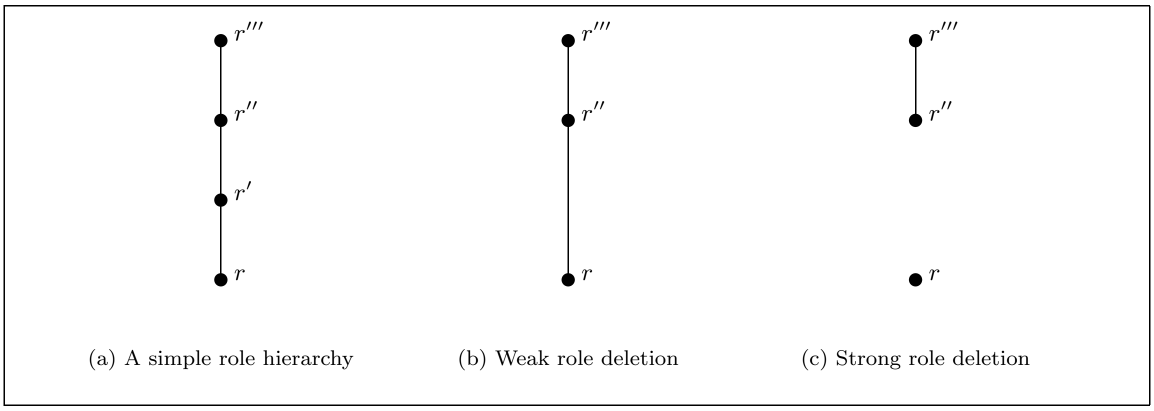


**Figure 8.3:** Role deletion

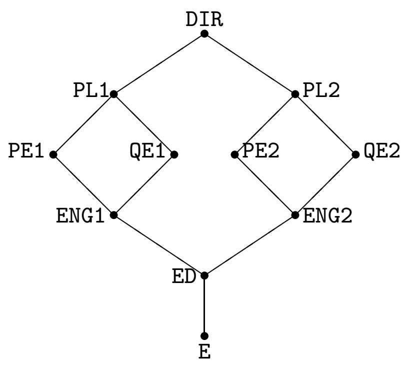
| RBAC96/ | SHRBAC | |
|---|---|---|
| ARBAC97 | Mandatory | Discretionary |
| $UA$ | $\pi_{ua}$ | $UA$ |
| $PA$ | $\pi_{pa}$ | $PA$ |
| `can-assign` | $\pi_{ua}$ | $RH$, `ua-constraints` |
| `can-revoke` | $\pi_{ua}$ | $RH$ |
| `can-assignp` | $\pi_{pa}$ | $RH$, `pa-constraints` |
| `can-revokep` | $\pi_{pa}$ | $RH$ |
| `can-modify` | $\pi_{rh}$ | $RH$ |

Table 8.1: A comparison of RBAC96 and SHRBAC features

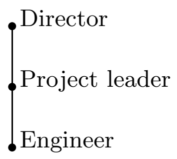
of his position, `bill` would be able to activate `PL1` and `PL2` (and any roles junior to them). However, $UA$ prevents `bill` from activating `PL2`. Similarly, the concept of administrative scope limits and distinguishes the administrative powers of roles with the same position. For example, $\phi(\texttt{PL1}) \succ \phi(\texttt{QE2})$; however, the administrative scope of `PL1` is $[\texttt{ENG1}, \texttt{PL1}]$, which prevents `PL1` making changes to the assignments of `QE2` or changing the hierarchy in the neighbourhood of `QE2`.

### 8.1.4 Requests

Requests consist of zero or more logical tests which we call the conditional part of the request and one or more operations. Operations update $S$, $RH$, $UA$, $PA$ or $V$. Recall that we denote the administrative scope of a role $a$ by $\sigma(a)$. For convenience we omit the current state $\langle RH, UA, PA, S, V\rangle$ from the parameter list in the definition of requests. We assume the existence of the data types `role`, `user`, `session`, `permission` and `antichain`. The set of commands $\Gamma$ that will be used in SHRBAC are given in Commands 8.1 – 8.14. We have commented the first command in some detail in order to make the intended semantics of the pseudo-code more explicit. Thereafter, we only comment where appropriate.



(a) Role hierarchy



(b) Position hierarchy

| $u$ | $\phi(u)$ |
|---|---|
| anne | {Engineer} |
| bill | {Project leader} |
| claire | {Director} |
| dave | {Engineer} |
| emma | {Engineer} |
| fred | {Engineer} |

(c) User-position associations

| $r$ | $\phi(r)$ |
|---|---|
| DIR | {Director} |
| PL1, PL2 | {Project leader} |
| PE1, PE2 | {Engineer} |
| QE1, QE2 | {Engineer} |
| ENG1, ENG2 | {Engineer} |
| ED | {Engineer} |
| E | $\emptyset$ |

(d) Role-position associations

| $u$ | $R(u)$ | $\downarrow R(u)$ |
|---|---|---|
| claire | {DIR} | {E, ED, ..., DIR} |
| bill | {PL1} | {E, ED, ENG1, PE1, QE1, PL1} |

(e) User-role assignments

**Figure 8.4:** An example of a SHRBAC system

**Command 8.1**

```
boolean assign-user($a$ role, $r$ role, $u$ user)
    if
        $\phi(a) \succ \phi(u)$ and                         /* $a$ is more senior than $u$ */
        $\phi(u) \succcurlyeq \phi(r)$ and                     /* $u$ is at least as senior as $r$ */
        $r \in \sigma(a)$                      /* $r$ is in the administrative scope of $a$ */
    then
        $UA = UA \cup \{(u,r)\}$                                       /* update $UA$ */
        assign-user = true                                  /* request succeeds */
    else
        assign-user = false                                    /* request fails */
```

**Command 8.2**

```
boolean revoke-user($a$ role, $r$ role, $u$ user)
    if
        $\phi(a) \succ \phi(u)$ and
        $r \in \sigma(a)$
    then
        $UA = UA \setminus \{(u,r)\}$
        revoke-user = true
    else
        revoke-user = false
```

**Command 8.3**

```
boolean assign-permission($a$ role, $r$ role, $p$ permission)
    if
        $\phi(a) \succ \phi(r)$ and
        $\phi(r) \succcurlyeq \phi(p)$ and
        $r \in \sigma(a)$
    then
        $PA = PA \cup \{(p,r)\}$
        assign-permission = true
    else
        assign-permission = false
```

**Command 8.4**

```
boolean revoke-permission($a$ role, $r$ role, $p$ permission)
    if
        $\phi(a) \succ \phi(r)$ and
        $r \in \sigma(a)$
    then
        $PA = PA \setminus \{(p,r)\}$
        revoke-permission = true
    else
        revoke-permission = false
```

**Command 8.5**

```
boolean start-session($u$ user, $i$ string, $R(s)$ antichain)
    if
        $R(s) \subseteq \downarrow R(u)$          /* $\downarrow R(u)$ is the set of roles assigned to $u$ */
    then
        $R(s) = \downarrow R(s)$                  /* form the down set of active roles */
        $S = S \cup \{(i,u,R(s))\}$
        start-session = true
    else
        start-session = false
```

**Command 8.6**

```
boolean stop-session($i$ string)
    $S = S \setminus \{(i, u, R(s))\}$
    stop-session = true
```

**Command 8.7**

```
boolean get-permission($(i, u, R(s))$ session, $p$ permission)
    if
        $R(p) \cap R(s) \neq \emptyset$
                                      /* $R(s)$ is the set of roles implicitly assigned to $s$ */
                             /* $R(p)$ is the set of roles to which $p$ is explicitly assigned */
    then
        $V = V \cup \{(i, u, p)\}$
        get-permission = true
    else
        get-permission = false
```

**Command 8.8**

```
boolean release-permission($i$ string, $u$ user, $p$ permission)
    $V = V \setminus \{(i, u, p)\}$
    release-permission = true
```

**Command 8.9**

```
boolean add-edge($a$ role, $r_p$ role, $r_c$ role)          /* $r_p$ is parent; $r_c$ is child */
    if
        $\phi(r_c) \preccurlyeq \phi(r_p)$ and
        $r_c \in \sigma(a)$ and
        $r_p \in \sigma(a)$ and
        $r_c \parallel r_p$              /* ensure no path or transitive edge is introduced */
    then
        $RH = RH \cup \{(r_c, r_p)\}$
        for all $r \in \Delta r_c$                                /* all children of $r_c$ */
            if $(r, r_p) \in RH$
                $RH = RH \setminus \{(r, r_p)\}$                  /* delete transitive edges */
        for all $r \in \nabla r_p$                                 /* all parents of $r_p$ */
            if $(r_c, r) \in RH$
                $RH = RH \setminus \{(r_c, r)\}$                  /* delete transitive edges */
        add-edge = true
    else
        add-edge = false
```

**Command 8.10**

```
boolean delete-edge($a$ role, $r_p$ role, $r_c$ role)            /* weak edge deletion */
                             /* we assume that $r_p$, $r_c$ $\notin$ $R(s)$ for any session $s$ */
    if
        $(r_c, r_p) \in RH$ and
        $r_c \in \sigma(a)$ and
        $r_p \in \sigma(a)$
    then
        for all $r \in \nabla r_p$        /* insert transitive edges that would be lost */
            $RH = RH \cup \{(r_c, r)\}$
        for all $r \in \Delta r_c$        /* insert transitive edges that would be lost */
            $RH = RH \cup \{(r, r_p)\}$
        $RH = RH \setminus \{(r_c, r_p)\}$                              /* delete edge */
        delete-edge = true
    else
        delete-edge = false
```

**Command 8.11**

```
boolean delete-edge-strong($a$ role, $r_p$ role, $r_c$ role)
                             /* we assume that $r_p$, $r_c$ $\notin$ $R(s)$ for any session $s$ */
    if
        $(r_c, r_p) \in RH$ and
        $r_c \in \sigma(a)$ and
        $r_p \in \sigma(a)$
    then
        $RH = RH \setminus \{(r_c, r_p)\}$
        delete-edge-strong = true
    else
        delete-edge-strong = false
```

**Command 8.12**

```
boolean add-role($a$ role, $r$ role, $\nabla r$ antichain, $\Delta r$ antichain, $p$ antichain)
     /* $r$ is the new role and will be assigned to the antichain of positions $p$ */
                                                  /* we assume that $\nabla r \neq \emptyset$ */
    if
        $\nabla r \subseteq \sigma(a)$ and
        $\Delta r \subseteq \sigma(a)$ and
        for all $r \in \nabla r$                                        /* preserve ...  */
            $p \preccurlyeq \phi(r)$
                and
        for all $r \in \Delta r$
            $p \not\succcurlyeq \phi(r)$                        /* ... hierarchy consistency */
                and
        for all $r' \in \nabla r$                            /* ensure no path or ...  */
            for all $r'' \in \Delta r$
                $r' \parallel r''$               /* ... transitive edge will be introduced */
    then
        for all $r' \in \nabla r$
            $RH = RH \cup \{(r, r')\}$                      /* connect $r$ to its parents */
        for all $r' \in \Delta r$
            $RH = RH \cup \{(r', r)\}$                     /* connect $r$ to its children */
        for all $r' \in \nabla r$
            for all $r'' \in \Delta r$
                if $(r'', r') \in RH$
                    $RH = RH \setminus \{(r'', r')\}$            /* delete transitive edges */
        $\phi(r) = p$                                                   /* update $\phi$ */
        add-role = true
    else
        add-role = false
```

**Command 8.13**

```
boolean delete-role($a$ role, $r$ role)                          /* weak role deletion */
    if
        $\nabla r \subseteq \sigma(a)$ and
        $\Delta r \subseteq \sigma(a)$
    then
        for all $r' \in \nabla r$
            for all $p \in P(r)$
                $PA = PA \setminus \{(p, r)\} \cup \{(p, r')\}$        /* re-assign permissions */
            $RH = RH \setminus \{(r, r')\}$
        for all $r' \in \Delta r$
            for all $u \in U(r)$
                $UA = UA \setminus \{(u, r)\} \cup \{(u, r')\}$              /* re-assign users */
            $RH = RH \setminus \{(r', r)\}$
        for all $r' \in \nabla r$
            for all $r'' \in \Delta r$
                $RH = RH \cup \{(r'', r')\}$                 /* insert transitive edges */
        delete-role = true
    else
        delete-role = false
```

**Command 8.14**

```
boolean delete-role-strong($a$ role, $r$ role)
    if
        $\nabla r \subseteq \sigma(a)$ and
        $\Delta r \subseteq \sigma(a)$
    then
        for all $r' \in \nabla r$
            for all $p \in P(r)$
                $PA = PA \setminus \{(p,r)\} \cup \{(p,r')\}$          /* re-assign permissions */
            $RH = RH \setminus \{(r,r')\}$
        for all $r' \in \Delta r$
            for all $u \in U(r)$
                $UA = UA \setminus \{(u,r)\} \cup \{(u,r')\}$                /* re-assign users */
            $RH = RH \setminus \{(r',r)\}$
        delete-role-strong = true
    else
        delete-role-strong = false
```

### 8.1.5 The basic security theorem for SHRBAC

**Theorem 8.1.1** *If the initial state, $t_0 = \langle RH_0, UA_0, PA_0, S_0, V_0\rangle$ of a SHRBAC system $\mathsf{S}(P, \Gamma, U, E, \phi, t_0)$ is secure and satisfies $\{\pi_{ra}, \pi_{pu}, \pi_{rh}\}$, then all possible evolutions of the system are secure and satisfy $\{\pi_{ra}, \pi_{pu}, \pi_{rh}\}$.*

**Proof** The proof proceeds by induction on the length of the evolution. In all cases, the base case is trivially satisfied by the assumption that $t_0$ satisfies $\{\pi_{ra}, \pi_{pu}, \pi_{rh}\}$.

We consider each of the security properties separately by examining the effect of each of the requests that can affect those properties. We assume that the evolution $t^N$ is secure and satisfies $\{\pi_{ra}, \pi_{pu}, \pi_{rh}\}$ (inductive hypothesis). (That is, all states $t_n$, where $n \leqslant N$, are secure and satisfy $\{\pi_{ra}, \pi_{pu}, \pi_{rh}\}$.)

The state $t_{N+1}$ is secure

*Proof* The only requests that change $V_N$ are `get-permission` and `release-permission`.

If `get-permission`$((i, u, R(s)), p)$ returns `true`, then $V_{N+1} = V_N \cup \{(i, u, p)\}$ and $R(s) \cap R(p)$ is non-empty. That is, there exists $r' \in R(s)$ such that $(p, r) \in PA$ and since $PA_N$ satisfies $\pi_{pu}$, $\phi(p) \preccurlyeq \phi(r')$. Now for all $s = (i, u, R(s)) \in S$, $R(s) \subseteq {\downarrow}R(u)$. Therefore, for all $r' \in R(s)$, there exists $r \in R(u)$ such that $r' \leqslant r$. Given that $t_N$ satisfies $\pi_{rh}$, we have $r' \leqslant r$ implies $\phi(r') \preccurlyeq \phi(r)$. Furthermore, since $t_N$ satisfies $\pi_{ra}$, $\phi(r) \preccurlyeq \phi(u)$. Therefore, if $V_N$ is secure and $V_{N+1} = V_N \cup \{(i, u, p)\}$, then $\phi(p) \preccurlyeq \phi(r') \preccurlyeq \phi(r) \preccurlyeq \phi(u)$ for some $r' \in R(s)$ and hence $V_{N+1}$ is secure.

If `release-permission` returns `true`, then $V_{N+1} = V_N \setminus \{(i, u, p)\}$. Clearly, by inductive hypothesis, $V_{N+1}$ is secure. $\square$

The state $t_{N+1}$ satisfies $\pi_{ra}$

*Proof* The only requests that change $UA_N$ are `assign-user`, `revoke-user`, `delete-role` and `delete-role-strong`.

If `assign-user` returns `true`, then $UA_{N+1} = UA_N \cup \{(u, r)\}$, where $\phi(u) \succcurlyeq \phi(r)$ (by construction of `assign-user`). Hence, by inductive hypothesis, $UA_{N+1}$ satisfies $\pi_{ra}$.

If `revoke-user` returns `true`, then $UA_{N+1} = UA_N \setminus \{(u, r)\}$. Clearly if $UA_N$ satisfies $\pi_{ra}$, then so does $UA_{N+1}$.

If `delete-role` returns `true`, then $UA_{N+1} = UA_N \setminus \{(u, r) : u \in U(r)\} \cup \{(u, r') : u \in U(r), r' \in \Delta r\}$. By inductive hypothesis, $\phi(u) \succcurlyeq \phi(r) \succcurlyeq \phi(r')$ for all $r' \in \Delta r$. Hence $UA_{N+1}$ satisfies $\pi_{ra}$. (A similar argument applies to `delete-role-strong`.)

The only requests that change $S_N$ are `start-session` and `stop-session`.

If `start-session` returns `true`, then $S_{N+1} = S_N \cup \{(i, u, R')\}$, where $R' \subseteq {\downarrow}R(u)$. By inductive hypothesis, $\phi(u) \succcurlyeq \phi(r)$ for all $r \in R'$. Hence $S_{N+1}$ satisfies $\pi_{ra}$.

If `stop-session` returns `true`, then $S_{N+1} = S_N \setminus \{(i, u, R')\}$. By inductive hypothesis, $S_{N+1}$ satisfies $\pi_{ra}$. $\square$

The state $t_{N+1}$ satisfies $\pi_{pu}$

*Proof* The only requests that change $PA_N$ are `assign-permission`, `revoke-permission`, `delete-role` and `delete-role-strong`.

If `assign-permission` returns `true`, then $PA_{N+1} = PA_N \cup \{(p, r)\}$, where $\phi(r) \succcurlyeq \phi(p)$ (by construction of `assign-permission`). Hence, by inductive hypothesis, $PA_{N+1}$ satisfies $\pi_{pu}$.

If `revoke-permission` returns `true`, then $PA_{N+1} = PA_N \setminus \{(p, r)\}$. Clearly if $PA_N$ satisfies $\pi_{pu}$, then so does $PA_{N+1}$.

If `delete-role` returns `true`, then $PA_{N+1} = PA_N \setminus \{(p, r) : p \in P(r)\} \cup \{(p, r') : p \in P(r), r' \in \nabla r\}$. By inductive hypothesis, $\phi(r') \succcurlyeq \phi(r) \succcurlyeq \phi(p)$ for all $r' \in \nabla r$. (A similar argument applies to `delete-role-strong`.) □

The state $t_{N+1}$ satisfies $\pi_{rh}$

*Proof* The only requests that change $RH$ are `add-edge`, `delete-edge`, `delete-edge-strong`, `add-role`, `delete-role` and `delete-role-strong`.

If `add-edge` returns `true`, then $RH_{N+1} = RH_N \cup \{(r_c, r_p)\} \setminus \{(r_c, r) : r \in \nabla r_p\} \setminus \{(r, r_p) : r \in \Delta r\}$, where $\phi(r_c) \preccurlyeq \phi(r_p)$. By inductive hypothesis $RH_{N+1}$ satisfies $\pi_{rh}$.

If `delete-edge` returns `true`, then $RH_{N+1} = RH_N \setminus \{(r_c, r_p)\} \cup \{(r_c, r) : r \in \nabla r_p\} \cup \{(r, r_p) : r \in \Delta r\}$. By inductive hypothesis, $\phi(r_c) \preccurlyeq \phi(r_p) \preccurlyeq \phi(r)$ for all $r \in \nabla r$, and $\phi(r) \preccurlyeq \phi(r_c) \preccurlyeq \phi(r_p)$, for all $r \in \Delta r$. Hence $RH_{N+1}$ satisfies $\pi_{rh}$.

If `delete-edge-strong` returns `true`, then $RH_{N+1} = RH_N \setminus \{(r_c, r_p)\}$. By inductive hypothesis $RH_{N+1}$ satisfies $\pi_{rh}$.

If `add-role` returns `true`, then $RH_{N+1} = RH_N \cup \{(r, r') : r' \in \nabla r\} \cup \{(r', r) : r' \in \Delta r\} \setminus \{(r', r'') : r' \in \Delta r, r'' \in \nabla r\}$, where $\phi(r) \preccurlyeq \phi(r')$ for all $r' \in \nabla r$ and $\phi(r) \succcurlyeq \phi(r')$ for all $r' \in \Delta r$. Hence, by inductive hypothesis $RH_{N+1}$ satisfies $\pi_{rh}$.

If `delete-role` returns `true`, then $RH_{N+1} = RH_N \setminus \{(r, r') : r' \in \nabla r\} \setminus \{(r', r) : r' \in \Delta r\} \cup \{(r', r'') : r' \in \Delta r, r'' \in \nabla r\}$. By inductive hypothesis, $\phi(r') \preccurlyeq \phi(r'')$ for all $r' \in \Delta r$ and $r'' \in \nabla r$. Hence $RH_{N+1}$ satisfies $\pi_{rh}$. (A similar argument applies to `delete-role-strong`.) □

■

We conclude this section by noting that we could have defined our requests differently. In particular, there is no reason why part of the conditional statement of `assign-user` should be that the parameters satisfy $\pi_{ra}$. (In fact, this would be analogous to the Bell-LaPadula model in which an access right can be added to the $ij$th element of the protection matrix even if $s_i$ and $o_j$ violate one or more of the security properties.) In this case, the definition of `get-permission` would need to be amended so that the conditional part confirms that $\phi(u) \succcurlyeq \phi(p)$. Our approach makes the assumption that the request `assign-user` would be executed far less frequently than `get-permission`, and that it therefore makes sense to incorporate the logical test in `assign-user`.

## 8.2 The secure hierarchical protection matrix model

In this section we examine the behaviour of models based on the SHA framework which employ a protection matrix reference monitor. A *secure hierarchical protection system* $\mathsf{S}(A, \Gamma, M_0, E, \phi, \Pi)$

is defined by the set of access rights, the set of requests, the initial protection matrix, the set of positions, the seniority function and the seniority policy. We will not present a single detailed model like SHRBAC. Instead, we will investigate how different choices of policy give rise to well known existing access control models. For example, it is clear that the protection matrix model is a special case of the secure hierarchical authorization framework in which there is no policy and no position hierarchy.

**Proposition 8.2.1** *The Bell-LaPadula model is a special case of the secure hierarchical authorization framework in which $E = C \cup K$ and the policy is $\{\pi_{ssp}, \pi_*\}$.*

**Proof** $C$ and $K$ are disjoint posets. $\mathcal{A}(C)$ is a chain, $C'$ say, since $C$ is a chain. In fact $C' = \{\emptyset, \{\mathtt{u}\}, \{\mathtt{c}\}, \{\mathtt{s}\}, \{\mathtt{t}\}\}$, and for all $c_1, c_2 \in C$, $c_1 < c_2$ implies $\{c_1\} \preccurlyeq \{c_2\}$. Furthermore, $\langle \mathcal{A}(K), \preccurlyeq \rangle = \langle 2^K, \subseteq \rangle$ since $K$ is an antichain. By Proposition 6.3.3,

$$\mathcal{A}(C \cup K) = \mathcal{A}(C) \times \mathcal{A}(K) = C' \times 2^K. \tag{8.1}$$

The security lattice in the Bell-LaPadula model (shown in Figure 2.6) is a sublattice of the one defined in (8.1). Furthermore, $\lambda(o) \leqslant \lambda(o')$ implies that $\phi(o) \preccurlyeq \phi(o')$. ■

The policy statements $\pi_{ssp}$ and $\pi_*$ are equivalent to the simple security property and the *-property, respectively. Hence we have a model that incorporates the ideas of the Bell-LaPadula model but permits a more flexible security lattice and information flow policy.

We now consider the simple example from Chapter 1 in which there are two departments, Payroll and Personnel to illustrate the use of the secure hierarchical protection matrix model. In each department there are Clerks and Administrators. We require that no Personnel staff can access Payroll information and that no Payroll staff can access Personnel information, with the exception of Payroll administrators who must be able to read (but not write) basic Personnel information such as national insurance numbers. The position hierarchy is shown in Figure 8.5. For all subjects $s$ associated with Payroll administrators, we define $\phi(u) = \{\mathtt{Payroll\ Admin}, \mathtt{Personnel\ Clerk}\}$. For all objects $o$ maintained by the Personnel department, we define $\phi(o) = \{\mathtt{Personnel\ Clerk}\}$. (We can of course refine Personnel staff and Payroll administrator access to those objects using the protection matrix.) The seniority of all other subjects is an appropriate singleton position.

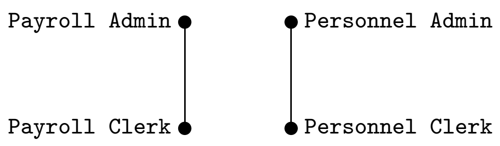


**Figure 8.5:** A simple position hierarchy

We note that the position hierarchy in this example can easily be represented as a sublattice of the usual Bell-LaPadula security lattice. Our intention is merely to demonstrate the use of our ideas not the extent to which they can be exploited.

In the remainder of this section we explore the models that arise as a result of defining different policy statements. For certain choices of policy we have a natural generalization of the typed access matrix model and also a means of defining transition secure commands (McLean 1994).

### 8.2.1 The typed access matrix model

The typed access matrix model (Sandhu 1992c) introduces the notion of *type* for subjects and objects in the protection matrix model. It extends the concept of entity types in the schematic protection model (Sandhu 1988) to the protection matrix model.

A *typed access matrix* (TAM) system $\mathsf{S}(A, \Gamma, M_0, T, \tau)$ consists of a finite set of commands $\Gamma$, a finite set of access rights $A$, a non-empty finite set of types $T$, an initial access matrix $M_0$, and a function $\tau : O_0 \to T$.[1] The body of each command contains primitive operations from the protection matrix model. A *create command* contains either a `create subject` or `create object` operation. We denote the set of create commands by $\Gamma_C \subseteq \Gamma$. Given a create command $\gamma$, $t \in T$ is a *child type* in $\gamma$ if there exists a formal parameter $s$ in $\gamma$ such that $\tau(s) = t$ and an operation of the form `create subject` $s$ or `create object` $s$; $t$ is a *parent type* in $\gamma$, otherwise. In Command 8.15, type `b` is a child type, while type `a` is both a parent and child type.[2]

**Command 8.15**

```
create-ab(a1:a,a2:a,b:b)             /* a1, a2 are of type a; b is of type b */
    create subject a2                                    /* a is a child type */
    create subject b                                     /* b is a child type */
    ...                                              /* additional operations */
```

The *creation graph* of a monotonic TAM (MTAM) system is a directed graph with vertex set $T$ and an edge from $t_1$ to $t_2$ if there is a create command in which $t_1$ is a parent type and $t_2$ is a child type. A MTAM system is *acyclic* if the creation graph is acyclic. (A MTAM system that included Command 8.15 would not be acyclic.) We define a binary relation $<$ on the set of create commands in the following way: $\gamma_1 < \gamma_2$ if a child type in $\gamma_1$ is a parent type in $\gamma_2$ or there is a directed path in the creation graph from a child type in $\gamma_1$ to a parent type in $\gamma_2$. Clearly $<$ is transitive, and if the creation graph is acyclic then $<$ is irreflexive and anti-symmetric.

A *canonical* TAM system is one in which all create commands are unconditional. Sandhu (1992c) showed that every TAM system is equivalent to a canonical TAM system. A MTAM system is *ternary* if no command has more than three parameters.

**Theorem 8.2.1 (Sandhu 1992c, Theorem 3)** *The safety problem for acyclic MTAM systems is* ***NP****-hard.*

**Proof** (Sketch) The proof of this theorem (and Theorem 8.2.2 below) requires that the TAM system be in *canonical form.* We topologically sort (Knuth 1973) the set of create commands (using the partial order induced by the parent and child types) to create a total order. We then proceed down the ordered list of commands and apply each command once to each possible tuple of parent entities. Sandhu (1992c) showed that this construction, the *unfolding* algorithm, terminates for acyclic authorization schemes. (This follows immediately from the fact that the initial set of entities and the set of commands are finite, and that the application of a create command cannot

[1]We use the term TAM system rather than the original *authorization scheme* (Sandhu 1992c) in order to keep our terminology consistent.

[2]The notation $s$ : `t` in the parameter list is used to denote that $s$ is of type `t`.

result in the creation of a type which could be used as a parent type in a command that has already been considered.) ■

**Theorem 8.2.2 (Sandhu 1992c, Theorem 5)** *The safety problem for ternary, acyclic MTAM systems is decidable in polynomial time in the size of the initial matrix.*

This result follows from the fact that there is an upper bound on the number of parameters in any command. Sandhu (1992c) also showed that ternary TAM systems have similar expressive power to the protection matrix model. Furthermore, binary TAM systems have less expressive power than ternary TAM systems. In short, ternary TAM systems provide the optimum combination of expressive power and low complexity.

The development of the TAM model was motivated in part by the ORGCON (originator controlled) policy, a US military policy that had been found to be difficult to implement within the Bell-LaPadula model (Abrams et al. 1991). The ORGCON policy requires that the creator (originator) of a document retains sole control over the propagation of access rights to that document.

Sandhu (1992c) demonstrates that the ORGCON policy can be implemented in the TAM model by ensuring that any subject that can access an ORGCON document is typed at the lowest level possible and therefore cannot copy the document or grant access (wittingly or unwittingly) to another subject. The ORGCON policy would normally be part of a wider (military) security policy and hence the typed access matrix would replace the usual protection matrix in the Bell-LaPadula model.

We claim the TAM model is an instance of the SHA framework; as such, it is easy to integrate it into the Bell-LaPadula model. We first define the *subject creation property* $\pi_{sc} \triangleq \phi(s_p) \succ \phi(s_c)$. A system that implements $\pi_{sc}$ must ensure that for each create command, every child subject in that command has a more senior parent subject. Given an acyclic TAM system $\mathsf{S}(A, \Gamma, M_0, T, \tau)$ we can construct the (acyclic) creation graph of that system. The transitive reduction of the creation graph is the position hierarchy $E$. The seniority function $\phi$ is identically equal to $\tau$. The parameter typing in commands in $\Gamma$ can be removed; create commands must now test the position of each parameter and ensure that the subject creation property is satisfied before creating any new subjects. (If we include the security labels and needs-to-know categories of the Bell-LaPadula model, the position hierarchy becomes $C \cup K \cup E$. In this case, $\phi(s) = \{\lambda(s), \tau(s)\}$, where $\lambda$ is the security function, and $\Pi = \{\pi_{ssp}, \pi_*, \pi_{sc}\}$.)

The advantage of the secure hierarchical protection matrix model is that given a position hierarchy, provided the commands implement the subject creation property correctly, the creation graph is necessarily acyclic. In the TAM model, there is no guarantee that the creation graph derived from a set of commands will be acyclic.

In the TAM model, a create command in which $t \in T$ is both a child and parent type is *attenuating* if any new subject of type $t$ has fewer capabilities than one of its parents of type $t$. An attenuating, acyclic TAM system is one in which the creation graph may contain loops (cycles of length 1), but is otherwise acyclic, provided each command that gives rise to a loop is attenuating. The safety problem for attenuating, acyclic MTAM systems remains decidable (Sandhu 1992c).

Attenuating, acyclic systems are simple to write in the secure hierarchical protection matrix model. We replace the subject creation property with the *weak subject creation property* $\pi'_{sc} \triangleq$

$\phi(s_p) \succcurlyeq \phi(s_c)$ and ensure that the entry of access right $a$ into $[s_c, o]$ can only occur if $a \in [s_p, o]$. Figure 8.16 shows an attenuating command.

**Command 8.16**

```
create-subject($s, s', o, p$)      /* $s$ is a parent subject; $s'$ is a child subject */
                                                        /* $p$ is the position of $s'$ */
    if
        $\phi(s) \succcurlyeq p$ and                                    /* $\pi_{tc'}$ is satisfied */
        $a$ in $[s, o]$                            /* check attenuating property */
    then
        create subject $s'$
        $\phi(s') = p$                                       /* assign position to $s'$ */
        enter $a$ in $[s', o]$
        ...                                                /* additional operations */
```

In addition, the deletion of matrix entries and the destruction of objects can be controlled using the respective positions of the relevant parameters. In particular, we can extend the notion of child and parent to *passive* and *active* objects in a command, where an object is passive if it is created, deleted, or changed by the command and active otherwise. The definition of the subject creation property can be extended so that for every passive object in a command there must be an active object that is more senior. In this way, we may be able to exercise more control over the behaviour of a non-monotonic secure hierarchical protection matrix system. An analysis of the safety problem for such systems is beyond the scope of this thesis.

### The typed access matrix model: An alternative formulation

We conclude our discussion of the TAM model with a simpler characterization of an acyclic TAM system. Given a TAM system $\mathsf{S}(A, \Gamma, M_0, T, \tau)$ we define the binary relation $\sim$ on $\Gamma_C$, where $\gamma_1 \sim \gamma_2$ if, and only if, a child type in $\gamma_1$ is a parent type in $\gamma_2$.

**Proposition 8.2.2** *The graph $\langle \Gamma_C, \sim \rangle$ is acyclic if, and only if, the creation graph of $\mathsf{S}$ is acyclic. Furthermore, $\gamma_1 < \gamma_2$ if, and only if, $\gamma_1 \sim^+ \gamma_2$, where $\sim^+$ denotes the transitive closure of $\sim$.*

**Proof** For each edge $(t_i, t_{i+1})$ in a path $(t_1, t_2), (t_2, t_3), \dots, (t_{n-1}, t_n)$ in the creation graph, there exists a create command $\gamma_i$ in which $t_i$ is a parent type and $t_{i+1}$ is a child type. We can view such a path in the following way:

$$\underbrace{(t_1, [t_2)}_{\gamma_1}, \underbrace{(t_2], [t_3)}_{\gamma_2}, \dots, \underbrace{(t_{i-1}, [t_i)}_{\gamma_{i-1}}, \underbrace{(t_i], t_{i+1})}_{\gamma_i}, \dots, \underbrace{(t_{n-2}, [t_{n-1})}_{\gamma_{n-2}}, \underbrace{(t_{n-1}], t_n)}_{\gamma_{n-1}},$$

where the square brackets indicate that commands $\gamma_{i-1}$ and $\gamma_i$ exist in which $t_i$ is a child and a parent type, respectively. Hence, for every such path in the creation graph, there exists an equivalent path $(\gamma_1, \gamma_2), \dots, (\gamma_{n-2}, \gamma_{n-1})$ in $\langle \Gamma_C, \sim \rangle$. In particular, the creation graph is acyclic if, and only if, the graph $\langle \Gamma_C, \sim \rangle$ is acyclic.

If $\gamma_1 < \gamma_2$, then, by definition, either $\gamma_1 \sim \gamma_2$ or there is a directed path in the creation graph from a child type in $\gamma_1$ to a parent type in $\gamma_2$. In the latter case, there also exists a directed path in $\langle \Gamma_C, \sim \rangle$, by the first part of this proposition. That is, $\gamma_1 \sim^+ \gamma_2$. ■

By Proposition 8.2.2, $\mathsf{S}$ is acyclic if $\langle \Gamma_C, \sim \rangle$ is acyclic. The unfolding algorithm can be applied directly to a topological ordering of $\langle \Gamma_C, \sim \rangle$. Hence this formulation does not require the construction of the creation graph or the ordering on $\Gamma_C$.

### 8.2.2 Transition secure systems

In Chapter 2 we briefly discussed the idea due to McLean of a function $\psi : S \cup O \to 2^S$, where $\psi(o)$ is the set of subjects that can change $\lambda(o)$. We now demonstrate how we can indirectly define $\psi$ using a policy statement. Recall that a command $\gamma$ is transition secure (McLean 1994) if, and only if, $\gamma(s, (V, M, \lambda)) = (V', M', \lambda')$ implies that for all $o$ such that $\lambda(o) \neq \lambda(o')$, $s \in \psi(o)$. Informally, the parameter list of a transition secure command is extended to include an "active" subject $s$ which can change the security clearance of an object subject to the conditions imposed by $\psi$. A simple way of implementing this requirement is to require that $\phi(s) \succ \phi(o)$. That is, $\psi(o) = \uparrow\phi(o) \setminus \{o\}$. In other words, the secure hierarchical protection matrix model provides a simple and natural definition for the function $\psi$.

McLean does not constrain the values of $\lambda(s)$ and $\lambda'(o)$. It seems reasonable to assume that $s$ should be at least as senior as the new seniority level of $o$. Therefore, we extend the scheme by defining the *transition secure property* $\pi_{ts} \triangleq (\phi(s) \succ \phi(o)) \wedge (\phi(s) \succcurlyeq \phi'(o))$. In other words, a command that implements $\pi_{ts}$ correctly ensures that the subject must be more senior than the object and at least as senior as the new seniority level of the object. Command 8.17 shows a command that implements $\pi_{ts}$.

**Command 8.17**

```
change-phi($s, s', p$)                              /* $s$ changes position of $s'$ to $p$ */
    if
        $\phi(s) \succ \phi(s')$ and
        $\phi(s) \succcurlyeq p$
    then
        $\phi(s') = p$                                      /* change position of $s'$ */
        ...                                         /* additional operations */
```

## 8.3 Discussion

We have presented a general framework for constructing access control models. Although the SHA framework has little practical use because of the generality of its features, it provides a starting point for developing access control models with well-defined security properties. In order to achieve this, a particular reference monitor and a specific policy will need to be chosen.

The SHRBAC model is one such instance of the SHA framework which employs role-based concepts. In particular, it uses the *RH*, *UA* and *PA* relations from RBAC96 and the SARBAC model for administrative purposes.

The most significant difference between SHRBAC and RBAC96/ARBAC97 is that all changes to the state of the system must satisfy requirements based on relative seniority in the position hierarchy. In particular, this prevents a user from being assigned a role that is too senior for that user's capabilities or responsibilities within the organization. This distinguishes SHRBAC from both RBAC96 and OASIS in which the legitimacy of a new user-role assignment is determined by existing user-role assignments. Furthermore, the availability of a permission to a user is controlled by the relative positions of the permission and the user as well as the usual permission-role assignment relation. In short, SHRBAC provides a model for systems with greater assurance than those based on the RBAC96 model. In particular, we note that the safety problem for role-based access control as defined in Chapter 5 is rendered trivial in SHRBAC. However, once we introduce requests that can change the position hierarchy and the position of entities in the system, greater care will need to be taken in order to ensure that this property is not compromised. (In this context, we are mindful of the work by McLean (1990) on the Bell-LaPadula model. However, we believe that the concept of administrative scope may well provide a natural way of controlling the behaviour of these requests.)

Unfortunately, the current version of SHRBAC incurs substantial administrative overheads. In particular, it is necessary to assign a position to every user and permission. One of the great advantages of RBAC96 is that it reduces the amount of access control management required. Furthermore, it is not immediately apparent whether all the security properties are required. The model is "defensive", in the sense that it requires that $\pi_{ra}$, $\pi_{pu}$ and $\pi_{rh}$ be satisfied. The requests would be less complex if we only require that a state is secure. In addition, we would not need to assign positions to roles. However, this results in a model in which every user and every permission could be assigned to DIR, for example. In mitigation, we observe that a similar situation arises in the Bell-LaPadula model when every entry in the protection matrix contains every access right.

There are more general difficulties with the SHA framework: it may be difficult to decide what the correct policy statements should be and to reason about the implications of the statements chosen; some applications may not be suitable or may require a more fine-grained specification of policy. With reference to the latter point, there has been considerable research into policy specification which supports more flexible access control policies (Damianou et al. 2000; Woo and Lam 1993). However, this research assumes that such policies will be written correctly. We believe that, in practice, it may be as difficult to write such policies, to reason about their correctness and to establish the subsequent behaviour of systems (the safety problem) as it is to describe the behaviour of protection systems. The great benefit of the SHA framework in this context is that, provided the position hierarchy is correctly specified, it is relatively easy to implement a set of requests that guarantee that the resulting system satisfies the order-based policy chosen. This discussion merely highlights one of the recurring themes in computer science: what is the best compromise between expressive power and complexity?

# Chapter 9

# Conclusions and Future Work

The broadest interpretation of the contributions of this thesis is that the application of simple mathematical models to role-based access control can make a significant contribution to the understanding of the way in which current role-based access control models operate. Furthermore, the use of existing and new mathematical results suggest extensions to existing role-based access control models that can improve their performance and utility.

More specifically, in Chapter 4, we introduced the RHA$_4$ and SARBAC models which provide a formal framework for role-based administration. These models are derived from a simple set of requirements and the notion of administrative scope of a role. A mathematical description of an encapsulated range, the basic unit of administration in the RRA97 model, provided the inspiration for the definition of administrative scope.

RHA$_4$ controls changes to the role hierarchy by assuming that such changes are made by an administrative role and are constrained by the administrative scope of that role. We believe that RHA$_4$ is a more comprehensive and comprehensible approach than that adopted in RRA97 and other models for administration of the role hierarchy.

SARBAC is a complete model for administration which extends the use of administrative scope to user- and permission-role assignments. We believe SARBAC is simpler, more intuitive and more coherent than ARBAC97. An interesting application of the SARBAC model is the development of an alternative approach to discretionary access control within a role-based framework. A further useful consequence of the SARBAC model is that the analysis of the behaviour of a role-based access control system, which is managed using SARBAC, is considerably simplified. An application of this is the analysis in Chapter 5 of the safety problem in RBAC96 using the SARBAC administrative model.

In Chapter 7 we applied the novel construction of a lattice of antichains to a simple model of conflict of interest policies, the lattice operations providing a natural way of composing two different policies. The simplicity of our approach enables us to investigate the intrinsic complexity of conflict of interest policies.

In Chapter 8 we described the secure hierarchical authorization framework, a second application of the lattice of antichains, which provides a template for the development of new access control models which incorporate an order-based security policy. These models constrain the propagation of access rights in accordance with the security policy and are intended to provide a basis for the design of systems in which greater assurances can be given about the security of the system. The models exhibit similar characteristics to the Bell-LaPadula model for multi-level secure systems. However, the secure hierarchical authorization framework assumes the existence of a supervision hierarchy which is intended to model the relative seniority of positions in an enterprise. The order-based security policy is framed in the context of this hierarchy, rather than the more rigid security lattice of the Bell-LaPadula model. The motivation for this additional flexibility is to provide models that have a wider applicability than the Bell-LaPadula model.

The SHRBAC model is a role-based access control model derived from the secure hierarchical authorization framework. Users, roles and permissions are assigned a level of seniority and user- and permission-role assignments are required to satisfy constraints (policy statements) expressed in terms of seniority. One useful feature of the model, which is absent from existing models, is that user-role assignments are limited by the seniority of the user. The most significant drawback of the model is its greater administrative and conceptual complexity compared to other role-based models. The current version of SHRBAC does not implement the complete SARBAC model and should therefore be regarded as work in progress. We discuss the development of the SHRBAC model and suggest a simpler approach to secure role-based models in Section 9.1.4.

In this thesis we have considered several issues in access control by examining their underlying mathematical structure: most importantly, administration in role-based access control models, secure access control models and conflict of interest policies. The results have been encouraging.

We believe that the concept of administrative scope, which has a purely mathematical characterization, could prove to be a valuable concept in role-based administration. We also believe that the RHA family of models and SARBAC model are a significant improvement on existing models for role-based administration.

The mathematical model for conflict of interest policies is extremely simple but retains the characteristic behaviour exhibited by more complex formulations such as RCL 2000. The focus on underlying structure rather than a specific application domain in our model suggests ways in which the specification of RCL 2000 can be strengthened. For example, conflicting sets should be antichains rather than arbitrary subsets. We have also identified simple yet practical applications of conflict of interest policies that have not previously been identified. We believe that the use of SARBAC together with the idea of a ceiling on the set of roles to which a user can be assigned provides considerable flexibility and control over the assignment of users to roles.

The mathematical analysis of the complexity of conflict of interest policies is of limited interest to computer scientists: the complexity of policies that contain constraints with no more than two elements is already unattractive enough computationally, without considering constraints with an arbitrary number of elements. The analysis is of limited interest to the mathematical community because Korshunov's theorem provides a good approximation for the number of Sperner families. Furthermore, we believe that Lemmas 7.3.2 and 7.3.3 may be special cases or corollaries of existing, albeit rather complex, results (Engel 1997, Section 2.3). It would seem that the more general analysis of lattices of antichains in Chapter 6, while not mathematically profound, is more original.

It is more difficult to assess the merit of the secure hierarchical authorization framework because a considerable number of issues require further work. Nevertheless, it is gratifying that several different existing access control models are instances of the framework.

## 9.1 Future work

A general criticism that could be levelled at the work in this thesis is the absence of any implementation of the theoretical material. Clearly, the most compelling argument for accepting a new model is that it works. Therefore, one priority of future work is the development of software that implements some of the ideas in this thesis. In addition, Chapters 4 – 8 provide many opportunities for future research, both in partial orders and access control.

### 9.1.1 Extensions to role-based access control

Chapter 4 provides a theoretical foundation for administration in role-based access control. It would be valuable to develop a prototype role-based access control mechanism based on the RBAC96 model. This prototype could be extended to incorporate two different sets of administrative functionality based on the ARBAC97 and SARBAC models, respectively, and could be used to assess their relative merits.

One issue that SARBAC does not address is the administration of separation of duty. The work in Chapter 7 suggests that a conflicting set in RCL 2000 can be regarded as an antichain in the role hierarchy. Hence, a conflicting set is identical in structure to a SARBAC constraint. This suggests that we can change conflicting sets in the same way as constraints in the `ua-constraints` and `pa-constraints` relations.

An implicit assumption in the RHA and SARBAC models is that a role hierarchy resembles a rooted tree or a forest of rooted trees (with the root at the top of the hierarchy). Little research has been conducted into the use and administration of role hierarchies that resemble rooted trees in which the root is at the bottom of the hierarchy.

An instance of such a hierarchy occurs when roles are used to assign permissions to a directory structure in which the number of users authorized to access a given directory is approximately inversely proportional to the depth of the directory in the directory tree. An example of such a situation is shown in Figure 9.1.[1] We can imagine that `/a/a`, for example, is a directory which can be accessed by a subset of the users who can access the (parent) directory `/a`. This corresponds naturally to making $r_{aa}$ a more senior role than $r_a$ and assigning the appropriate users to $r_{aa}$. However, under the assumption that there is no “super user”, the role hierarchy in Figure 9.1b bears little resemblance to the role hierarchies examined in the literature (and this thesis). I was recently made aware of this problem by Dave Cohen of Royal Holloway.

It is encouraging to note that SARBAC can administer this hierarchy if we define an administrative role $a$ and define the extended hierarchy shown in Figure 9.1b. Then $a$ has no permissions to access the directories, but can assign users and permissions to these roles. (We can create encapsulated ranges in the role hierarchy of Figure 9.1b by adding a top and bottom element to the hierarchy. However, the addition of practically any role or edge – the creation of the role $r_{abc}$ as a parent of $r_{ab}$, for example – is not possible in ARBAC97.) We hope to undertake a more systematic study of these types of hierarchy and the suitability of ARBAC97 and SARBAC as administrative tools in these cases.

[1]Assignment to role $r_x$ implies access to directory `x`.

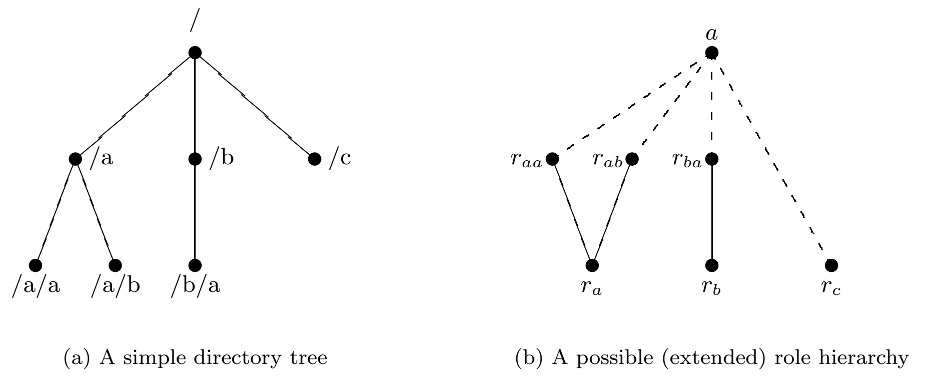

/
/a
/b
/c
/a/a
/a/b
/b/a
(a) A simple directory tree
$a$
$r_{aa}$
$r_{ab}$
$r_{ba}$
$r_a$
$r_b$
$r_c$
(b) A possible (extended) role hierarchy

**Figure 9.1:** Assigning permissions to a directory structure using a role hierarchy

We must also examine the properties of a line manager and whether the line manager of a role is a useful concept in role-based administration. In particular, role hierarchy operations generally involve more than one role. Therefore, we need to establish a definition of line manager for a set of roles before the concept is of real value. (The set of administrators of a single role is known to be a set with a unique minimal element by Proposition 4.2.3.) There are two plausible definitions of a line manager of a set of roles $R' \subseteq R$. Firstly, we view it as an extension of the concept of line manager for a single role and define the line manager of $R'$ to be the least upper bound of the set of line managers of the roles in $R'$. Unfortunately, in an arbitrary poset, this is not necessarily uniquely defined. Therefore, we may need to impose certain restrictions on role hierarchies. The most obvious such restriction is to insist that the (finite) extended role hierarchy is a (complete) lattice.

Secondly, we define the set of administrators of $R'$ to be the intersection of the administrators of each role. It is not immediately obvious what conditions the role hierarchy must satisfy to guarantee that the resulting set has a unique minimal element.

There may well be standard results in partial order theory which supply an answer to the above questions. (Indeed, it may be that existing results show that the two definitions of line manager given above are equivalent.)

Unlike $\text{RHA}_1$ and $\text{RHA}_2$, the concept of line manager does not arise naturally in $\text{RHA}_3$ because of the `admin-authority` relation. An obvious line of research is to investigate what constraint(s) must be applied to the `admin-authority` relation in order to recover this useful intuitive concept. Our intuition is that a suitable constraint would be that the role field is unique in `admin-authority`, the interpretation being that a role cannot be controlled by more than one administrative role. (In short, if the `admin-authority` relation is a function, then we claim that every role has a line manager.)

An increasingly important subject in access control is *delegation* (Barka and Sandhu 2000; Bandmann et al. 2001; Moffett and Sloman 1991). The concept of line manager may well prove useful here. For example, all functions could be delegated to the appropriate line manager; these functions could not then be performed by a more senior entity. Our limited understanding of current research in this area prevents a more thorough discussion of how this topic can be developed.

The use of $\text{DRBAC}_1$, role activation rules and a permission usage role hierarchy was briefly discussed in Chapter 4. The synthesis of ideas from this thesis, OASIS and ERBAC, respectively, may produce a role-based model that exhibits some useful behaviour and features. We are particularly interested in incorporating environmental and temporal constraints (which may be used in OASIS role activation rules) into `ua-constraints` and `pa-constraints`.

### 9.1.2 The safety problem in role-based access control

Chapter 5 represents the first steps in the analysis of the safety problem in role-based access control. The natural extension of this work is to attempt to characterize role-based access control systems in which the safety problem is decidable and, ideally, tractable. (Henceforth, we will call these decidable systems or tractable systems, as appropriate.) Clearly, the commands used in Sections 5.2.2 and 5.2.3 are not sufficiently restrictive to guarantee the decidability of the safety problem. It is also apparent that these commands are artificial; we would not expect to find such commands in an implementation of a role-based access control model. Nevertheless, the analysis in Chapter 5, like the earlier work of Harrison et al. (1976), suggests that we should not take the controlled propagation of access rights for granted.

We believe that describing useful, decidable role-based systems should be easier than for protection systems, because in the former we have the concept of administrative scope which necessarily limits the behaviour of commands. In a protection system, the administration of the protection matrix is governed by entries in the matrix itself. Administrative scope, however, is defined by the extended role hierarchy. Hence, if we can identify a class of extended hierarchies that do not permit the creation of arbitrarily long antichains in the hierarchy, then this may represent a class of decidable role-based systems (subject to appropriate constraints on the structure of commands).

Taking the obvious parallel from the study of the safety problem in the protection matrix model, we can also consider what it means for a set of commands to be monotonic and what impact this has on the safety problem. A monotonic role-based command would not permit the revocation and assignment of a permission (or user) in the same command. A study of the safety problem in the protection matrix model suggests that such a condition will not be sufficient to guarantee that the safety problem is decidable.

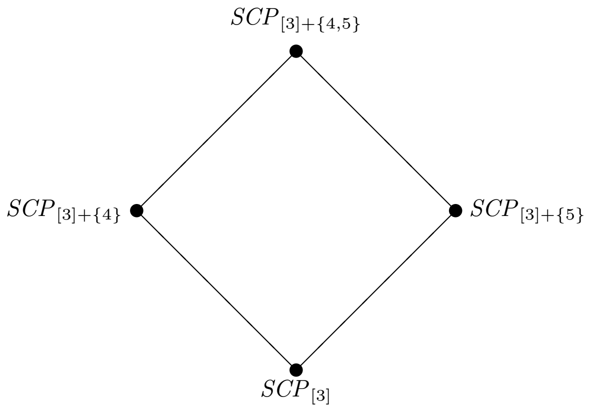
The use of the line manager concept may also have an impact on the safety problem. In particular, it has the effect of localizing the changes made to the role hierarchy. This bears a similarity to the behaviour of mono-conditional protection systems (Harrison and Ruzzo 1978), for which the safety problem is decidable. Hence, an interesting line of research will be to establish whether the safety problem is indeed decidable for RBAC96/SARBAC with this simple constraint on role hierarchy operations. If the answer is affirmative, it will be somewhat counter-intuitive because many different attempts have been made to find decidable cases of **SP:HRU**. Such an answer would also provide a significant reason for deploying role-based access control more widely.

### 9.1.3 Partial orders and symmetric chain partitions

The first problem we will consider is the (limiting) behaviour of a "$2^m$-symmetric chain partition" of $2^{[n]}$. For example, we can represent $SCP_n$ in terms of $2^2$ symmetric chain partitions that are copies of $SCP_{n-2}$. A 4-symmetric chain partition of $2^{[5]}$ is shown in Figure 9.2.

$$\begin{array}{cc}
\emptyset \lessdot \{1\} \lessdot \{1,2\} \lessdot \{1,2,3\} & \{4\} \lessdot \{1,4\} \lessdot \{1,2,4\} \lessdot \{1,2,3,4\} \\
\{2\} \lessdot \{2,3\} & \{2,4\} \lessdot \{2,3,4\} \\
\{3\} \lessdot \{1,3\} & \{3,4\} \lessdot \{1,3,4\} \\
 & \\
\{5\} \lessdot \{1,5\} \lessdot \{1,2,5\} \lessdot \{1,2,3,5\} & \{4,5\} \lessdot \{1,4,5\} \lessdot \{1,2,4,5\} \lessdot \{1,2,3,4,5\} \\
\{2,5\} \lessdot \{2,3,5\} & \{2,4,5\} \lessdot \{2,3,4,5\} \\
\{3,5\} \lessdot \{1,3,5\} & \{3,4,5\} \lessdot \{1,3,4,5\}
\end{array}$$

(a)



(b)

**Figure 9.2:** A 4-symmetric chain partition of $2^{[5]}$ induced by the set [3]

In fact, we can show that $X \subseteq [n]$, where $|X| = m \leqslant n$, "induces" a $2^{n-m}$-symmetric chain partition of $2^{[n]}$. We denote $[n] \setminus X$ by $X^C$. Consider the relation $\sim$ defined on $2^{[n]}$, where $Y \sim Z$ if, and only if, $Y \cap X^C = Z \cap X^C = \chi$, for some $\chi \subseteq X^C$. It is easy to verify that $\sim$ is an equivalence relation and that there exists a bijection from $2^X$ to each equivalence class of the form $Y \mapsto Y \cup \chi$, where $\chi \subseteq X^C$; $\chi$ is called the *characteristic set* of the equivalence class. Let $2^{[n]}/2^X$ denote the set of equivalence classes and let $2^{X+\chi} \in 2^{[n]}/2^X$ denote the equivalence class with characteristic set $\chi \subseteq X^C$. For $2^{X+\chi_1}, 2^{X+\chi_2} \in 2^{[n]}/2^X$, $2^{X+\chi_1} \leqslant 2^{X+\chi_2}$ if, and only if, $\chi_1 \subseteq \chi_2$.

In Figure 9.2a, $X = [3]$ and the characteristic sets of the four equivalence classes are $\emptyset, \{4\}, \{4,5\}, \{5\}$ (reading clockwise from the top left class). Figure 9.2b shows the poset $\langle 2^{[n]}/2^X, \leqslant \rangle$.

**Proposition 9.1.1** *For all $n \geqslant 0$, $X \subseteq [n]$, $\langle 2^{[n]}/2^X, \leqslant \rangle \cong \langle 2^{[k]}, \subseteq \rangle$, where $k = n - |X|$.*

**Proof** (Sketch) It can easily be verified that the function $\zeta : 2^{[k]} \to 2^{[n]}/2^X$, where $\zeta(Y) = 2^{X+Y}$, is the required order isomorphism. ■

We can represent $2^X$ as a symmetric chain partition, and extend the symmetric chain partition to each element of $2^{[n]}/2^X$ in an obvious way (as shown in Figure 9.2a). We will denote a symmetric chain partition of $X$ by $SCP_X$ and call a chain in $SCP_X$ a *base* chain. Then for every base chain $\mathcal{C} \in SCP_X$, there exists a *corresponding* chain in $SCP_{X+X}$.

**Proposition 9.1.2** *Given a $2^k$-symmetric chain partition of $2^{[n]}$, the number of choices from a set of corresponding chains each of length 2 is $|\mathcal{A}_{k+1}|$; the number of choices from a set of corresponding chains each of length 1 is $|\mathcal{A}_k|$.*

**Proof** (Sketch) Consideration of Figure 9.2a shows that the elements of a set of corresponding chains of length 2 can be arranged as a lattice isomorphic to $2^{[k+1]}$. Similarly, the elements of a set of corresponding chains of length 1 can be arranged as a lattice isomorphic to $2^{[k]}$. The result follows. ■

If we consider the $2^k$-symmetric chain partition induced by a set $X$ of size $m = n - k$, we see that each component of the partition is itself a symmetric chain partition of $m$ elements and hence contains $\binom{m}{\lfloor m/2 \rfloor}$ chains. Hence we have the following conjecture.

**Conjecture 9.1.1** *For all $k < n$,*

$$|\mathcal{A}_n| \leqslant |\mathcal{A}_{k+1}|^{\binom{n-k}{\lfloor (n-k)/2 \rfloor}}.$$

In particular, the case $k = 0$ corresponds to a symmetric chain partition; by Conjecture 9.1.1 we have

$$|\mathcal{A}_n| \leqslant |\mathcal{A}_1|^{\binom{n}{\lfloor n/2 \rfloor}} = 3^{\binom{n}{\lfloor n/2 \rfloor}},$$

confirming Hansel's result (Theorem 2.2.3). The case $k = 1$ corresponds to a bi-symmetric chain partition; by Conjecture 9.1.1,

$$|\mathcal{A}_n| \leqslant |\mathcal{A}_2|^{\binom{n-1}{\lfloor (n-1)/2 \rfloor}} = 6^{\binom{n-1}{\lfloor (n-1)/2 \rfloor}},$$

which corresponds to Theorem 7.3.2. (The case $k = n - 1$ yields $|\mathcal{A}_n| \leqslant |\mathcal{A}_n|^{\binom{1}{0}} = |\mathcal{A}_n|$.) We note that in order to prove the conjecture, it will be necessary to prove that every filter is counted at least once and that choices made from longer chains do not contradict choices made earlier in the construction (as we did in the proof of Theorem 7.3.2).

The second problem concerns finding upper and lower bounds for the number of conflict of interest policies in an arbitrary partial order $X$, which we will denote $|\mathcal{A}_X|$. This question is of interest in view of the comments made in Remark 7.2.1 about conflict of interest policies in a role-based access control environment, where the access control context (in this case, the set of roles) is a partially ordered set. For example, given the role hierarchy in Figure 7.2, $|R| = 3$ and $|\mathcal{A}_R| = 10$, whereas $|\mathcal{A}_3| = 19$. One way of tackling this problem may be to generalize the definition of symmetric chain partition to an arbitrary poset $X$ and thereby produce an upper bound for $|\mathcal{A}_X|$. It is immediately obvious that $|\mathcal{A}_X| \geqslant 2^{|X|}$ since for all $Y \subseteq X$, $\bigcup_{y \in Y} \{y\}$ is a conflict of interest policy in $X$.

A further potential area of research is the use of the lattice operations in $\langle \mathcal{A}_n, \preccurlyeq \rangle$ to define crossover operations in grouping genetic algorithms (Falkenauer 1998). A grouping genetic algorithm can be used to find a partition of a set of variables. This partition must also satisfy certain requirements which are expressed as a fitness function. A potential solution (partition) is represented as a sequence of genes or *chromosome*. A chromosome with a high level of fitness represents a good approximate solution to the grouping problem.

A *crossover operation* takes two chromosomes with high fitness and combines them to create two children, the intention being to create offspring that inherit the characteristics of the parents that led to their fitness. (Chromosomes are also subject to random mutation in order to extend the search space.) It is difficult to find crossover operations for genetic grouping algorithms that create two children that exhibit the characteristics of their parents (Tucker 2001).

Trivially, a partition is a Sperner family in the powerset of the variable set. Hence, the binary operations of $\langle \mathcal{A}_n, \preccurlyeq \rangle$ and $\langle \mathcal{A}_n, \preccurlyeq' \rangle$ can be used to create two new Sperner families. The resulting Sperner families are not necessarily partitions. However, removing duplicates in one of two ways chosen at random leads to two child partitions that retain features of their parents.

For example, given a set of 10 variables and the parent chromosomes

$$\begin{array}{|c||c|c||c|c|c||c|c|c|c|}\hline 2 & 9 & 7 & 5 & 4 & 3 & 6 & 8 & 1 & 10 \\ \hline \end{array}$$
$$\begin{array}{|c||c||c|c||c|c||c|c|c|c|}\hline 8 & 4 & 1 & 6 & 2 & 7 & 5 & 3 & 10 & 9 \\ \hline \end{array},$$

we can use the binary operation $\wedge'$ to create a child chromosome.[2] That is, we form the union of the parents to obtain

$$\begin{array}{|c||c||c||c|c||c|c||c|c||c|c|c||c|c|c|c||c|c|c|c|}\hline 2 & 8 & 4 & 1 & 6 & 2 & 7 & 9 & 7 & 5 & 4 & 3 & 6 & 8 & 1 & 10 & 5 & 3 & 10 & 9 \\ \hline \end{array},$$

and then delete all supersets to obtain

$$\begin{array}{|c||c||c||c|c||c|c||c|c|c|c|}\hline 2 & 8 & 4 & 1 & 6 & 9 & 7 & 5 & 3 & 10 & 9 \\ \hline \end{array}.$$

Finally we delete all repeated elements (working from left to right, for example) to obtain

$$\begin{array}{|c||c||c||c|c||c|c||c|c|c|}\hline 2 & 8 & 4 & 1 & 6 & 9 & 7 & 5 & 3 & 10 \\ \hline \end{array}.$$

"Sperner crossover" operations appear to produce children that more accurately reflect their ancestry than existing crossover operations. This should accelerate the convergence of the grouping genetic algorithm to good solutions. However, the representation of the chromosome used to implement Sperner crossover operations efficiently is more complicated and increases the overheads of the mutation process. Hence, we intend to conduct experiments to see whether these operations accelerate the convergence of a grouping genetic algorithm sufficiently to offset the increased computational overheads.

### 9.1.4 Secure access control models

The work of McLean showed that a secure system as defined by Bell-LaPadula does not necessarily imply that the system exhibits secure behaviour in any practical sense. This arises if commands

[2]The subsets comprising the partition are delimited by bold vertical lines and are assumed to be in order of increasing cardinality.

can update both the seniority function and the contents of the protection matrix. (Tranquillity means that the seniority function cannot be changed and the problems identified by McLean do not arise.) There is an analogy to be drawn here between monotonic and non-monotonic protection systems. In the absence of tranquillity, if the level of a subject is increased, say, then the potential authorizations of that subject change. In particular, the subject will (subject to suitable entries in the protection matrix) acquire read access to objects at the new security level but will lose write access to objects at the old security level.

A natural extension to the work of Chapter 8 is to construct a set of commands for a secure hierarchical protection matrix model which includes the simple security property, the *-property and the transition secure property $\pi_{ts}$ and either state and prove a suitable security theorem or examine the complexity of the safety problem.

The SHRBAC model requires additional work in terms of administration. We need to incorporate the `admin-authority`, `ua-constraints` and `pa-constraints` as (discretionary) features of the model and also consider how separation of duty constraints can be included and administered. An alternative to `ua-constraints` and `pa-constraints` would be to use the ordering $\preccurlyeq'$ on the lattice of antichains. This ordering can be used to define minimal rather than maximal requirements for access to an object. For example, we could say that a user $u$ can be assigned to a role $r$ provided $\phi(R(u)) \succcurlyeq' \phi(r)$.

We do not currently consider modelling changes to the position hierarchy. Our experience with SARBAC suggests that administering the position hierarchy should not be too problematic. We must also provide requests that can create new users and objects and that can change the seniority function. We believe that the administrative scope function and a suitable security property would be sufficient to realize the latter feature.

We discussed some drawbacks to SHRBAC in Chapter 8. We note that a much simpler model can be developed directly from RBAC96, which, for convenience, we will refer to as MLSRBAC. Let us assume that all permissions are "charged" and can be either *up*, *down* or *neutral*. The (set of) *effective roles* with respect to a permission $p$, denoted $R_E(p)$, is defined as follows:

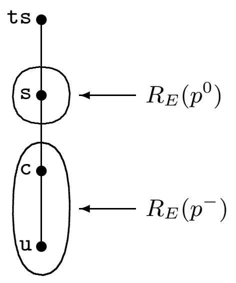
$$R_E(p) = \begin{cases} \uparrow R(p) & \text{if } p \text{ is an up permission,} \\ \downarrow R(p) & \text{if } p \text{ is a down permission,} \\ R(p) & \text{if } p \text{ is a neutral permission.} \end{cases}$$

We also assume that a session is modelled as an antichain of roles $A$ and is a subset of the roles implicitly assigned to the user running the session. However, the permissions of the roles in $\downarrow A$ are not available to the user. Instead, a request by a session for permission $p$ is granted if $A \cap R_E(p) \neq \emptyset$. Additionally, administrative role $a$ can assign $p$ to $r$ if either $p$ is an up permission and $r \in \sigma(a)$ or $p$ is a down permission and $\downarrow r \subseteq \sigma(a)$ or $p$ is a neutral permission and $r \in \sigma(a)$.

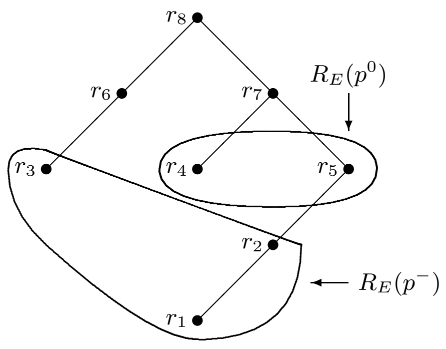
The intuition here is that the *UA* relation acts as a security function, where up permissions correspond to read-type access rights in the Bell-LaPadula model and down permissions correspond to write-type access rights. (Neutral permissions can be regarded as read-write-type access rights.) Figure 9.3 shows two MLSRBAC hierarchies. Figure 9.3a illustrates the close correspondence between MLSRBAC and the Bell-LaPadula model. For example, assume that the down permission

$p^- = (o, \texttt{append})$ is assigned to the "role" `c`, the neutral permission $p^0 = (o, \texttt{write})$ is assigned to `s`, and that `bill` is assigned to `ts`.[3] Then `bill` can `append` to $o$ only if he activates a session using either the `u` or `c` role, and can `write` to $o$ only if he activates a session using `s`. A user assigned to `c` would not be able to `write` to $o$.

Let us suppose now that $p^-$ is assigned to $r_2$ and $r_3$ in the hierarchy in Figure 9.3b, and that $p^0$ is assigned to $r_5$. In addition, suppose that `bill` is assigned to $r_6$ and $r_7$. Then `bill` can `append` to $o$ only if he activates a session using at least one of the roles $r_1$, $r_2$ and $r_3$.



(a) A Bell-LaPadula "role hierarchy"



(b) A general role hierarchy

**Figure 9.3:** MLSRBAC hierarchies

MLSRBAC has a number of potential advantages. Firstly, the intended semantics of inheritance in the role hierarchy becomes explicit: the use of roles in a session is determined solely by the activation hierarchy; the availability of permissions to a session is determined solely by the usage hierarchy. Secondly, it provides a natural way of simulating multi-level secure systems with a single hierarchy, hence making it simpler than existing approaches to this topic. Thirdly, it extends the class of hierarchies that can be used to implement multi-level secure properties. This, in turn, means it is possible to define "multi-level secure" hierarchies that correspond more closely to enterprise characteristics. Finally, it would seem from the examples based on the hierarchies in Figure 9.3 that the principle of least privilege is accurately modelled in MLSRBAC. (The overloading of inheritance in an RBAC96 hierarchy and the uni-directional flow of permission inheritance in the hierarchy means that the principle of least privilege is not necessarily adhered to in RBAC96.) Clearly, a more formal approach is required to confirm that MLSRBAC exhibits such advantages and to estimate the complexity of the model.

However, there are two questions that also require further research. Firstly, the example in Figure 9.3b raises an awkward question: Is it correct to insist that the set of active roles in a MLSRBAC session is an antichain? We can see that if `bill` wants $p^0$ and $p^-$ available in a session he must activate $r_1$ (or $r_2$) and $r_5$, which do not form an antichain. Therefore, we should consider whether this has a detrimental effect on the implementation and computational complexity of MLSRBAC. (Recall that we could assume in other role-based access control models that the set of

[3]Recall that `write` is a read and write access mode in the Bell-LaPadula model. We also note that there is no requirement for $o$ itself to have a particular security level.

roles in a session is an antichain, although it should be noted that RBAC96 does not insist on this condition.) Secondly, the analogy between MLSRBAC and the Bell-LaPadula model breaks down if we assume that the permission-assignment relation defines the "security level" of a permission. For example, $p^-$ is assigned to $r_2$ and $r_3$ and is available to a session that includes at least one of these roles. In the Bell-LaPadula model, it would be required that the session include *both* these roles. Therefore, we must consider whether these features of MLSRBAC compromise our ability to reason about the security properties of the MLSRBAC model.

Finally, we note that a natural area for further work is to investigate the interaction of ML-SRBAC and SARBAC. Given that MLSRBAC is essentially a constrained version of RBAC96 in which permissions can behave in more complex ways, we expect that much of this work will be straightforward and hope to report substantial progress in the near future.

# Notation

The symbols $\triangleq$ and $:=$ denote “is defined to be” and “is assigned the value”, respectively. The end of a proof is marked by $\blacksquare$; the end of a proof of a claim within a longer proof is marked by $\square$.

**Sets**

| | | |
|---|---|---|
| $[n]$ | the first $n$ non-zero natural numbers | 22 |
| $\mathbb{N}$ | the set of natural numbers | 22 |
| $\mathbb{N}^+$ | the set of non-zero natural numbers | 22 |
| $2^X$ | the powerset of $X$ | 22 |
| $\|X\|$ | cardinality of set $X$ | 22 |
| $X \dot\cup Y$ | disjoint union of $X$ and $Y$ | 26 |
| $X \times Y$ | cartesian product of $X$ and $Y$ | 26 |
| $\setminus$ | set difference | 38 |
| $\mathbb{R}$ | the set of real numbers | 38 |
| $\mathbb{Z}$ | the set of integers | 40 |
| $X^C$ | set complement of $X$ | 179 |

**Posets**

| | | |
|---|---|---|
| $\langle X, \leqslant \rangle$ | partially ordered set $X$ | 21 |
| $x < y$ | $x$ is strictly less than $y$ | 22 |
| $x \geqslant y$ | $x$ is greater than $y$ | 22 |
| $x \parallel y$ | $x$ is incomparable to $y$ ($\{x, y\}$ is an antichain) | 22 |
| $x \lessdot y$ | $x$ is covered by $y$ | 22 |
| $\Delta$ | lower shadow | 22 |
| $\nabla$ | upper shadow | 22 |
| $\mathcal{A}(X)$ | lattice of antichains in $X$ | 22 |
| $w(X)$ | width of $X$ | 23 |
| $Y^u$ | set of upper bounds of $Y$ | 23 |
| $Y^l$ | set of lower bounds of $Y$ | 23 |
| inf | greatest lower bound | 23 |
| sup | least upper bound | 23 |
| $x \vee y$ | least upper bound of $x$ and $y$ (join operation) | 24 |
| $x \wedge y$ | greatest lower bound of $x$ and $y$ (meet operation) | 24 |
| $\hookrightarrow$ | order-embedding | 24 |
| $L_1 \cong L_2$ | $L_1$ is isomorphic to $L_2$ | 24 |
| $\downarrow Y$ | “down” $Y$ | 24 |

| | | |
|---|---|---|
| $\downarrow y$ | "down" $\{y\}$ | 24 |
| $\uparrow Y$ | "up" $Y$ | 24 |
| $\uparrow y$ | "up" $\{y\}$ | 24 |
| $\mathcal{F}(X)$ | lattice of filters in $X$ | 24 |
| $\mathcal{I}(X)$ | lattice of ideals in $X$ | 24 |
| $(x, y)$ | open range | 25 |
| $[x, y]$ | closed range | 25 |
| $\mathcal{J}(L)$ | set of join-irreducible elements in $L$ | 25 |
| $X^{\partial}$ | dual of $X$ | 26 |
| $X \oplus Y$ | linear sum of $X$ and $Y$ | 26 |
| $\langle \mathcal{A}(X), \preccurlyeq \rangle$ | lattice of antichains in $X$ | 113 |
| $\underline{Y}$ | set of minimal elements in $Y$ | 115 |
| $\overline{Y}$ | set of maximal elements in $Y$ | 115 |
| $\langle \mathcal{A}(X), \preccurlyeq' \rangle$ | lattice of antichains in $X$ | 117 |
| $DM(X)$ | Dedekind-MacNeille completion of $X$ | 120 |

**Role-based access control**

| | | |
|---|---|---|
| $PA$ | permission-role assignment relation | 45 |
| $UA$ | user-role assignment relation | 45 |
| $RH$ | role hierarchy | 45 |
| $ARH$ | administrative role hierarchy | 46 |
| $P(r)$ | permissions assigned explicitly to $r$ | 47 |
| $\downarrow P(r)$ | permissions assigned implicitly and explicitly to $r$ | 47 |
| $R(p)$ | roles to which permission $p$ is assigned explicitly | 47 |
| $\uparrow R(p)$ | roles to which permission $p$ is assigned implicitly and explicitly | 47 |
| $R(u)$ | roles to which user $u$ is assigned explicitly | 47 |
| $\downarrow R(u)$ | roles to which user $u$ is assigned implicitly and explicitly | 47 |
| $U(r)$ | users assigned explicitly to $r$ | 47 |
| $\uparrow U(r)$ | users assigned implicitly and explicitly to $r$ | 47 |
| `AO` | `allother` function | 50 |
| `OE` | `oneelement` function | 50 |
| $(r)$ | immediate authority range of $r$ | 59 |
| $\sigma(r)$ | administrative scope of $r$ | 76 |
| $\sigma^{+}(r)$ | proper administrative scope of $r$ | 77 |
| $C(a)$ | set of roles controlled by $a$ | 80 |
| $\bigwedge R$ | conjunction of roles in $R$ | 86 |
| $\overline{R}$ | maximal elements in $R$ | 86 |
| $\underline{R}$ | minimal elements in $R$ | 86 |
| $\pi_{ra}$ | role activation property | 155 |
| $\pi_{pu}$ | permission usage property | 155 |
| $\pi_{rh}$ | hierarchy consistency property | 155 |
| $\pi_{ua}$ | user assignment property | 155 |
| $\pi'_{ua}$ | weak user assignment property | 155 |
| $\pi_{pa}$ | permission assignment property | 155 |
| $\pi'_{pa}$ | weak permission assignment property | 156 |

| | | |
|---|---|---|
| $R_E(p)$ | set of effective roles with respect to $p$ | 182 |

**Conflict of interest policies**

| | | |
|---|---|---|
| $SCP_n$ | symmetric chain partition of $2^{[n]}$ | 26 |
| $\binom{n}{r}$ | binomial coefficient | 27 |
| $\mathcal{A}_n$ | set of conflict of interest policies | 126 |
| $\mathcal{E}(\mathcal{P})$ | set of environments satsifying $\mathcal{P}$ | 125 |
| $\Sigma(\mathcal{P})$ | length of conflict of interest policy $\mathcal{P}$ | 133 |
| $\mathcal{A}_{n,r}$ | set of default maximal $r$-policies | 141 |
| $\begin{bmatrix} n \\ r \end{bmatrix}$ | maximal size of member of $\mathcal{A}_{n,r}$ | 141 |
| $\begin{vmatrix} n \\ r \end{vmatrix}$ | | 142 |
| $BCP_n$ | bi-symmetric chain partition of $[n]$ | 143 |
| $\nu(n)$ | | 147 |
| $\mu(n)$ | | 147 |
| $\mathcal{A}_n^2$ | | 149 |
| $2^{[n]}/2^X$ | | 179 |
| $2^{X+\chi}$ | | 179 |

**Miscellaneous**

| | | |
|---|---|---|
| $\lfloor x \rfloor$ | greatest integer less than or equal to $x$ (floor of $x$) | 27 |
| $\mathcal{O}(\phi(n))$ | Ooh ("big O") of $\phi(n)$ | 38 |
| $\forall$ | universal quantifier | 51 |
| $\lceil x \rceil$ | smallest integer greater than or equal to $x$ (ceiling of $x$) | 133 |

# Glossary of Security Terms

The terms in this glossary are widely used in the access control community and were introduced without definition in Chapter 1. We felt that "in-line" definitions would have been unnecessarily formal and would have made the chapter more cumbersome to read. The purpose of this glossary is to make explicit what we mean by those terms so that no ambiguity can arise in the mind of the reader. It is not a comprehensive list of terminology in the thesis. The definitions are mainly taken from Stoneburger (2000); two other useful sources are Saltzer and Schroeder (1975) and US Department of Defense (1988).

**access control**
: Enabling authorized use of a resource while preventing unauthorized use or use in an unauthorized manner.

**access control mechanism**
: The security engineering term for information system functionality that (1) controls all access, (2) cannot be by-passed, (3) is tamper-resistant, and (4) provides confidence that the preceding three items are true (also referred to as *reference monitor*).

**access control policy**
: The statement of authorization for information objects.

**accountability**
: The security goal requiring that the actions of an *entity* may be traced uniquely to that entity. This supports non-repudiation, deterrence, fault isolation, intrusion detection and prevention, and after-action recovery and legal action.

**assurance**
: Grounds for confidence that the other four security goals (*integrity*, *availability*, *confidentiality*, and *accountability*) have been adequately met by a specific implementation. "Adequately met" includes (1) functionality that performs correctly, (2) sufficient protection against unintentional errors (by users or software), and (3) sufficient resistance to malicious penetration or by-pass.

**authentication**
: Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in a system.

**authorization**
: The granting or denying of access rights to a user, program, or process.

**availability**
: The security goal requiring that intentional or accidental attempts to (1) perform unauthorized deletion of data or (2) otherwise cause a denial of service or data are prevented.

**confidentiality**
The security goal requiring that intentional or accidental attempts to perform unauthorized data reads are prevented. Like integrity, confidentiality covers data in storage, during processing, and while in transit.

**data integrity**
The property that data has not been altered in an unauthorized manner. Data integrity covers data in storage, during processing, and while in transit.

**denial of service**
The prevention of authorized access to resources or the delaying of time-critical operations.

**entity**
Either a *subject* or an *object*.

**group**
A set of users.

**information flow policy**
A *rule-based security policy* in which the rules relate the sensitivity of the objects and the subjects requesting access to those objects.

**integrity**
The security goal requiring that intentional or accidental attempts to violate *data integrity* or *system integrity* are prevented.

**information system security architecture**
A description of security principles and an overall approach for complying with the principles that drive the system design; that is, guidelines on the placement and implementation of specific security services within various distributed computing environments.

**object**
A passive *entity* that contains or receives information.

**rule-based security policy**
A security policy based on global rules imposed for all subjects.

**subject**
An active *entity*, generally in the form of a person, process, or device, that causes information to flow among objects or changes the system state.

**system integrity**
The property that a system has when it performs its intended function in the intended manner, free from unauthorized manipulation.

# References

Abadi, M., M. Burrows, B. Lampson, and G. Plotkin (1993). A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems 15*(4), 706–734.

Abrams, M., J. Heaney, O. King, L. LaPadula, M. Lazear, and I. Olson (1991). Generalized framework for access control: Towards prototyping the ORGCON policy. In *Proceedings of 14th National Computer Security Conference*, pp. 257–266.

Ahn, G.-J. and R. Sandhu (1999). The RSL99 language for role-based separation of duty constraints. In *Proceedings of Fourth ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 43–54.

Ahn, G.-J. and R. Sandhu (2000). Role-based authorization constraints specification. *ACM Transactions on Information and System Security 3*(4), 207–226.

Aho, A. and J. Ullman (1992). *Foundations of Computer Science*. New York: W.H. Freeman and Company.

Anderson, J. (1973). Information security in a multi-user computer environment. In *Advances in Computers*, Volume 12, pp. 1–35. New York: Academic Press.

Bacon, J., K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri (2000). Generic support for distributed applications. *IEEE Computer 33*(3), 68–76.

Bandmann, O., M. Dam, and B. Sadighi Firozabadi (2001). Constrained delegation. Preprint.

Barka, E. and R. Sandhu (2000). Framework for role-based delegation models. In *Proceedings of 16th Annual Computer Security Applications Conference*, New Orleans, Louisiana, pp. 168–177.

Barkley, J. (1997). Comparing simple role-based access control models and access control lists. In *Proceedings of Second ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 127–132.

Barkley, J., A. Cincotta, D. Ferraiolo, S. Gavrila, and D. Kuhn (1997). Role based access control for the World Wide Web. In *Proceedings of 20th National Computer Security Conference*, Baltimore, Maryland, pp. 331–340.

Bell, D. and L. LaPadula (1973a). Secure computer systems: Mathematical foundations. Technical Report MTR-2547, Volume I, Mitre Corporation, Bedford, Massachusetts.

Bell, D. and L. LaPadula (1973b). Secure computer systems: A mathematical model. Technical Report MTR-2547, Volume II, Mitre Corporation, Bedford, Massachusetts.

Bell, D. and L. LaPadula (1976). Secure computer systems: Unified exposition and Multics interpretation. Technical Report MTR-2997, Mitre Corporation, Bedford, Massachusetts.

Bertino, E., E. Ferrari, and V. Atluri (1999). The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security 2*(1), 65–104.

Biba, K. (1977). Integrity considerations for secure computer systems. Technical Report MTR-3153, Mitre Corporation, Bedford, Massachusetts.

Birkhoff, G. (1933). On the combination of subalgebras. *Proceedings of the Cambridge Philosophical Society 29*, 441–464.

Birkhoff, G. (1948). *Lattice Theory.* New York: American Mathematical Society.

Brualdi, R. (1999). *Introductory Combinatorics.* New Jersey: Prentice Hall.

Budd, T. A. (1983). Safety in grammatical protection systems. *International Journal of Computer and Information Sciences 12*(6), 413–431.

Burris, S. and H. Sankappanavar (1981). *A Course in Universal Algebra.* New York: Springer-Verlag.

Chen, F. and R. Sandhu (1995). Constraints for role-based access control. In *Proceedings of First ACM Workshop on Role-Based Access Control*, Gaithersburg, Maryland, pp. II39–II46.

Crampton, J. and G. Loizou (2000). Two partial orders on the set of antichains. Technical Report BBKCS-00-05, Birkbeck College, University of London, United Kingdom.

Crampton, J. and G. Loizou (2001a). Authorisation and antichains. *ACM Operating Systems Review 35*(3), 6–15.

Crampton, J. and G. Loizou (2001b). The completion of a poset in a lattice of antichains. *International Mathematical Journal 1*(3), 223–238.

Crampton, J. and G. Loizou (2001c). Role-based access control: The safety problem. Submitted.

Crampton, J., G. Loizou, and G. O'Shea (1999). Evaluating and improving access control. Technical Report BBKCS-99-11, Birkbeck College, University of London, United Kingdom.

Crampton, J., G. Loizou, and G. O'Shea (2001). A logic of access control. *The Computer Journal 44*(2), 137–149.

Damianou, N., E. Lupu, N. Dulay, and M. Sloman (2000). Ponder: A language for specifying security and management policies for distributed systems. Technical Report DOC 2000/1, Imperial College, University of London, United Kingdom.

Davey, B. and H. Priestley (1990). *Introduction to Lattices and Order.* Cambridge, United Kingdom: Cambridge University Press.

Dedekind, R. (1897). Über Zerlegungen von Zahlen durch ihre grössten gemeinsamen Teiler. In *Festschrift der Technische Hochschule Braunschweig bei Gelegenheit der 69. Versammlung deutscher Naturforscher und Ärzte*, pp. 1–40.

Demetrovics, J. (1978). On the number of candidate keys. *Information Processing Letters 7*, 266–269.

Denning, D. (1976). A lattice model of secure information flow. *Communications of the ACM 19*(5), 236–243.

Engel, K. (1997). *Sperner Theory.* Cambridge, United Kingdom: Cambridge University Press.

Falkenauer, E. (1998). *Genetic Algorithms and Grouping Problems.* Chichester, United Kingdom: Jonn Wiley & Sons.

Feinstein, H., R. Sandhu, C. Youman, and E. Coyne (1995). Final report: Small business innovation research (SBIR): Role-based access control: Phase I. Technical report, SETA Corporation, McLean, Virginia.

Ferraiolo, D. and J. Barkley (1997). Specifying and managing role-based access control within a corporate intranet. In *Proceedings of Second ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 77–82.

Ferraiolo, D., J. Barkley, and D. Kuhn (1999). A role based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security 2*(1), 34–64.

Ferraiolo, D., J. Cugini, and D. Kuhn (1995). Role-based access control (RBAC): Features and motivations. In *Proceedings of 11th Annual Computer Security Applications Conference*, New Orleans, Louisiana, pp. 241–248.

Ferraiolo, D. and D. Kuhn (1992). Role-based access control. In *Proceedings of the 15th NIST-NSA National Computer Security Conference*, Baltimore, Maryland, pp. 554–563.

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

Friberg, C. and A. Held (1997). Support for discretionary role-based access control in ACL-oriented operating systems. In *Proceedings of Second ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 83–93.

Garey, M. and D. Johnson (1979). *Computers and Intractability.* San Francisco, California: W.H. Freeman and Company.

Gavrila, S. and J. Barkley (1998). Formal specification for role based access control user/role and role/role relationship management. In *Proceedings of Third ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 81–90.

Gligor, V. (1995). Characteristics of role-based access control. In *Proceedings of First ACM Workshop on Role-Based Access Control*, Gaithersburg, Maryland, pp. II9–II14.

Gligor, V., S. Gavrila, and D. Ferraiolo (1998). On the formal definition of separation-of-duty policies and their composition. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, Oakland, California, pp. 172–183.

Goh, C. and A. Baldwin (1998). Towards a more complete model of role. In *Proceedings of Third ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 55–61.

Graham, G. and P. Denning (1972). Protection – principles and practice. In *Proceedings of the AFIPS Spring Joint Computer Conference*, Atlantic City, New Jersey, pp. 417–429.

Grätzer, G. (1978). *General Lattice Theory.* London, United Kingdom: Academic Press.

Greenlaw, R. and H. Hoover (1998). *Fundamentals of the Theory of Computation: Principles and Practice.* San Francisco, California: Morgan Kaufman.

Hansel, G. (1966). Sur le nombre des fonctions Booléennes monotones de $n$ variables. *Comptes Rendus de l'Académie des Sciences, Série A 262*, 1088–1090.

Harrison, M. and W. Ruzzo (1978). Monotonic protection systems. In R. DeMillo, D. Dobkin, A. Jones, and R. Lipton (Eds.), *Foundations of Secure Computation*, pp. 337–363. New York: Academic Press.

Harrison, M., W. Ruzzo, and J. Ullman (1976). Protection in operating systems. *Communications of the ACM 19*(8), 461–471.

Hayton, R. (1996). *Oasis, An Open Architecture for Secure Interworking Services.* Ph. D. thesis, University of Cambridge, Cambridge, United Kingdom. Technical Report 399.

Hayton, R., J. Bacon, and K. Moody (1998). Access control in an open distributed environment. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, California, pp. 3–14.

Heydon, A., M. Maimone, J. Tygar, J. Wing, and A. Zaremski (1989). Miró: Visual specification of security. Technical Report CMU-CS-89-1989, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Hine, J., W. Yao, J. Bacon, and K. Moody (2000). An architecture for distributed OASIS services. In *Proceedings of Middleware 2000*, Volume 1795 of *Lecture Notes in Computer Science*, Hudson River Valley, New York, pp. 107–123.

Hopcroft, J. and J. Ullman (1969). *Formal Languages and their Relation to Automata.* Reading, Massachusetts: Addison-Wesley.

Hua, L. and S. Osborn (1998). Modeling UNIX access control with a role graph. In *Proceedings of International Conference on Computers and Information*, Winnepeg, Manitoba, Canada.

Jajodia, S., P. Samarati, and E. Bertino (1997). A logical language for expressing authorizations. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, California, pp. 31–42.

Joshi, J., A. Ghafoor, W. Aref, and E. Spafford (2001). Digital government security infrastructure design challenges. *IEEE Computer 34*(2), 66–72.

Kleitman, D. and J. Makowsky (1975). On Dedekind's problem: The number of isotone boolean functions, II. *Transactions of the American Mathematical Society 213*, 373–390.

Knuth, D. (1973). *The Art of Computer Programming: Fundamental Algorithms* (2nd ed.). Reading, Massachusetts: Addison-Wesley.

Koch, G. and K. Loney (1997). *Oracle8: The Complete Reference.* Berkeley, California: Osborne/McGraw-Hill.

Korshunov, A. (1980). The number of monotone Boolean functions. *Problemy Kibernet. 38*, 5–108. In Russian.

Lampson, B. (1971). Protection. In *Proceedings of the Fifth Princeton Symposium on Information Sciences and Systems*, pp. 437–443. Reprinted in *ACM Operating Systems Review*, *8*(1):18–24,1974.

Lipton, R. and L. Snyder (1977). A linear time algorithm for deciding subject security. *Journal of the Association for Computing Machinery 24*(3), 455–464.

Lipton, R. and L. Snyder (1978). On synchronization and security. In R. DeMillo, D. Dobkin, A. Jones, and R. Lipton (Eds.), *Foundations of Secure Computation*, pp. 367–385. New York: Academic Press.

Lubell, D. (1966). A short proof of Sperner's lemma. *Journal of Combinatorial Theory 1*, 299.

Lupu, E., D. Marriott, M. Sloman, and N. Yialelis (1995). A policy based role framework for access control. In *Proceedings of First ACM Workshop on Role-Based Access Control*, Gaithersburg, Maryland, pp. II15–II24.

MacNeille, H. (1937). Partially ordered sets. *Transactions of the American Mathematical Society 42*, 416–460.

Martin, J. C. (1990). *Introduction to Languages and the Theory of Computation.* New York: McGraw-Hill.

McLean, J. (1990). The specification and modeling of computer security. *IEEE Computer 23*(1), 9–16.

McLean, J. (1994). Security models. In J. Marciniak (Ed.), *Encyclopedia of Software Engineering.* John Wiley & Sons.

Meshalkin, L. (1963). A generalization of Sperner's theorem on the number of subsets of a finite set. *Teoriya Veroyatnosteĭ ee Primeneniya 8*, 219–220. In Russian.

Minsky, M. (1967). *Computation: Finite and Infinite Machines.* Englewood Cliffs, New Jersey: Prentice Hall International Inc.

Moffett, J. (1998). Control principles and role hierarchies. In *Proceedings of Third ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 63–69.

Moffett, J. and E. Lupu (1999). The uses of role hierarchies in access control. In *Proceedings of Fourth ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 153–160.

Moffett, J. and M. Sloman (1991). Delegation of authority. In *Integrated Network Management II*, pp. 595–606. Elsevier North-Holland.

Mohammed, I. and D. Ditts (1994). Design for dynamic user role-based security. *Computers and Security 13*(8), 661–671.

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

Nyanchama, M. and S. Osborn (1993). Role-based security, object-oriented databases and separation of duty. *SIGMOD Record 22*(4), 45–51.

Nyanchama, M. and S. Osborn (1994). Access rights administration in role-based security systems. In J. Biskup, M. Morgenstern, and C. Landwehr (Eds.), *Database Security VIII: Status & Prospects*, pp. 37–56. Elsevier North-Holland.

Nyanchama, M. and S. Osborn (1995). The role graph model. In *Proceedings of First ACM Workshop on Role-Based Access Control*, Gaithersburg, Maryland, pp. II25–II31.

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

Osborn, S., R. Sandhu, and Q. Munawer (2000). Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security 3*(2), 85–106.

O'Shea, G. (1995). Redundant access rights. *Computers & Security 14*, 323–348.

O'Shea, G. (1997). *Access Control in Operating Systems*. Ph. D. thesis, Birkbeck College, University of London, United Kingdom.

Pittelli, P. (1987). The Bell-LaPadula computer security model represented as a special case of the Harrison-Ruzzo-Ullman model. In *Proceedings of the* $10^{th}$ *National Computer Security Conference*, Gaithersburg, Maryland, pp. 118–121.

Post, E. (1946). A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society 52*, 264–268.

Rabin, M. and J. Tygar (1987). An integrated toolkit for operating system security. Technical Report TR-05-87, Aiken Computer Laboratory, Harvard University, Cambridge, Massachusetts.

Sadighi Firozabadi, B. and M. Sergot (1999). Power and permissions in security systems. In B. Christianson, B. Crispo, J. Malcolm, and M. Roe (Eds.), *Proceedings of 7th International Workshop on Security Protocols*, Cambridge, United Kingdom, pp. 48–59.

Sadighi Firozabadi, B., M. Sergot, and O. Bandmann (2001). Using authority certificates to create management structures. In *Proceedings of 9th International Workshop on Security Protocols*, Cambridge, United Kingdom. To appear.

Saltzer, J. and M. Schroeder (1975). The protection of information in computer systems. *Proceedings of the IEEE 36*(9), 1278–1308.

Sandhu, R. (1988). The schematic protection model: Its definition and analysis for acyclic attenuation schemes. *Journal of the Association for Computing Machinery 35*(2), 404–432.

Sandhu, R. (1992a). Expressive power of the schematic protection model. *Journal of Computer Security 1*(1), 59–98.

Sandhu, R. (1992b). Lattice-based enforcement of Chinese Walls. *Computers & Security 11*(8), 753–763.

Sandhu, R. (1992c). The typed access matrix model. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, California, pp. 122–136.

Sandhu, R. (1992d). Undecidability of safety for the schematic protection model with cyclic creates. *Journal of Computer and System Sciences 44*(1), 141–159.

Sandhu, R. (1995). Workshop summary. In *Proceedings of First ACM Workshop on Role-Based Access Control*, Gaithersburg, Maryland, pp. I1–I7.

Sandhu, R. (1998). Role activation hierarchies. In *Proceedings of Third ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 33–40.

Sandhu, R., V. Bhamidipati, E. Coyne, S. Ganta, and C. Youman (1997). The ARBAC97 model for role-based administration of roles: Preliminary description and outline. In *Proceedings of Second ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 41–49.

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

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

Sandhu, R., D. Ferraiolo, and D. Kuhn (2000). The NIST model for role-based access control: Towards a unified standard. In *Proceedings of Fifth ACM Workshop on Role-Based Access Control*, Phoenix, Arizona, pp. 47–63. `http://www.acm.org/sigsac/nist.pdf`.

Sandhu, R. and Q. Munawer (1998a). How to do discretionary access control using roles. In *Proceedings of Third ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 47–54.

Sandhu, R. and Q. Munawer (1998b). The RRA97 model for role-based administration of role hierarchies. In *Proceedings of 14th Annual Computer Security Applications Conference*, Scottsdale, Arizona, pp. 39–49.

Sandhu, R. and J. Park (1998). Decentralized user-role assignment for Web-based intranets. In *Proceedings of Third ACM Workshop on Role-Based Access Control*, Fairfax, Virginia, pp. 1–12.

Simon, R. and M. Zurko (1997). Separation of duty in role-based environments. In *Proceedings of 10th IEEE Computer Security Foundations Workshop*, Rockport, Massachusetts, pp. 183–194.

Sloman, M. (1994). Policy driven management for distributed systems. *Journal of Network and Systems Management 2*(4), 333–360.

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

Sperner, E. (1928). Ein Satz über Untermengen einer endlichen Menge. *Mathematische Zeitschrift 27*, 544–548.

Stoneburger, G. (2000). Information system security engineering principles – initial draft outline. http://csrc.nist.gov/publications/drafts/issep-071800.doc.

Ting, T., S. Demurjian, and M. Hu (1992). Requirements capabilities and functionalities of user-role based security for an object-oriented design model. In C. Landwehr and S. Jajodia (Eds.), *Database Security V: Status & Prospects*, pp. 275–296. Elsevier North-Holland.

Tucker, A. (2001). *The Automatic Explanation of Multivariate Time Series.* Ph. D. thesis, Birkbeck College, University of London, United Kingdom.

Turing, A. (1936). On computable numbers with an application to the Entscheidungproblem. *Proceedings of the London Mathematical Society, Series 2 42*, 230–265.

US Department of Defense (1988). Glossary of computer security terms. Technical Report NCSC-TG-004-88, Department of Defense Computer Security Centre, Fort Meade, Maryland.

von Solms, S. and I. van der Merwe (1994). The management of computer security profiles using a role-oriented approach. *Computers and Security 13*(8), 673–680.

Woo, T. and S. Lam (1993). Authorization in distributed systems: A new approach. *Journal of Computer Security 2*(2-3), 107–136.

Yamamoto, K. (1954). Logarithmic order of free distributive lattices. *Journal of the Mathematical Society of Japan 6*, 343–353.

Yao, W., J. Bacon, and K. Moody (2001). A role-based access control model for supporting active security in OASIS. In *Proceedings of Sixth ACM Symposium on Access Control Models and Technologies*, Chantilly, Virginia, pp. 171–181.

Yialelis, N. and M. Sloman (1996). A security framework supporting domain based access control in distributed systems. In *IEEE Proceedings of the Internet Society Symposium on Network and Distributed Systems Security*, San Diego, California, pp. 26–39.

# Index