COMPARING THE EXPRESSIVE POWER OF ACCESS CONTROL MODELS

by Mahesh V. Tripunitara, Ninghui Li

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Comparing the Expressive Power of Access Control Models

Mahesh V. Tripunitara and Ninghui Li
Center for Education and Research in Information Assurance and Security
and Department of Computer Sciences
Purdue University
656 Oval Drive, West Lafayette, IN 47907-2086
{mtripuni, ninghui}@cs.purdue.edu

Abstract

Comparing the expressive power of access control models is recognized as a fundamental problem in computer security. Such comparisons are generally based on simulations between different access control schemes. However, the definitions for simulations that are used in the literature make it impossible to put results and claims about the expressive power of access control models into a single context. Furthermore, some definitions for simulations used in the literature such as those used for comparing RBAC (Role-Based Access Control) with other models, are too weak to distinguish access control models from one another in a meaningful way.

We propose a theory for comparing the expressive power of access control models. We perceive access control systems as state-transition systems and require simulations to preserve security properties. We discuss the rationale behind such a theory, apply the theory to reexamine some existing work on the expressive power of access control models in the literature and present three results. We show that: (1) RBAC with a particular administrative model from the literature (ARBAC97) is limited in its expressive power; (2) ATAM (Augmented Typed Access Matrix) is more expressive than TAM (Typed Access Matrix), thereby solving an open problem posed in the literature; and (3) a trust-management language is at least as expressive as RBAC with a particular administrative model (the URA97 component of ARBAC97).

1 Introduction

An access control system enforces a policy on who may access what resources and in what manner. Policies are generally expressed in terms of the current state of the system, and states that may result from prospective changes (e.g., “Alice should always have read access to a particular file, f”). Thus, when an access control system is perceived as a state-transition system, it consists of a set of states, rules on how state-transitions may occur and a set of properties or queries that are of interest in a given state (e.g., “Does Alice have read access to a particular file, f?”) Policies may then be expressed in terms of these components, and such policies may be verified to hold notwithstanding the fact that state-transitions occur.

An access control system is an instance of an access control scheme: a scheme specifies the types of state-transition rules that may be specified in a system based on that scheme. A set of access control schemes is an access control model. An example of an access control model is the access matrix model [5]. An example of a scheme based on the access matrix model is the HRU scheme [6] which specifies that state-transition rules are commands of a particular form. A specific set of HRU commands together with a start state is an example of an access control system. The expressive power of an access control model captures the notion of whether different policies can be represented in systems based on schemes from that model.

Comparing the expressive power of access control models is recognized as a fundamental problem in information security and is studied extensively in the literature [1, 3, 4, 15, 17, 18, 19]. The expressive power of a model is tied to the expressive power of the schemes from the model. In comparing schemes based on expressive power, we ask what types of policies can be represented by systems based on a scheme. If all policies that can be represented in scheme B can be represented in scheme A, then scheme A is at least as expressive as scheme B.

A common methodology used for comparing access control models in previous work is *simulation.* When a scheme A is simulated in a scheme B, each system in A is mapped to a corresponding system in B. If every scheme in one model can be simulated by some scheme in another model, then the latter model is considered to be at least as expressive as the former. Furthermore, if there exists a scheme in the latter model that cannot be simulated by any scheme in the former, then the latter model is strictly more expressive than the former. Different definitions for simulations are used in the literature on comparing access control models. We identify two axes along which these definitions differ.

- The first axis is whether a simulation is required to preserve safety properties. In the comparison of different schemes based on the access matrix model [1, 4, 18, 19], the preservation of safety properties is required. If a scheme A is simulated in a scheme B, then a system in scheme A reaches an unsafe state if and only if the image of the system under the simulation (which is a system in scheme B) reaches an unsafe state.

  On the other hand, the preservation of safety properties is not required in the simulations used for comparing MAC (Mandatary Access Control), DAC (Discretionary Access Control), and RBAC (Role-Based Access Control) [15, 17, 13]. Nor is it required in the simulations used for the comparison of Access Control Lists (ACL), Capabilities, and Trust Management (TM) systems [3]. In these comparisons, the requirement for a simulation of A in B is that it should be possible to use an implementation of the scheme B to implement the scheme A. We call this the *implementation paradigm* of simulations.

- The second axis is whether to restrict the number of state-transitions that the simulating scheme needs to make in order to simulate one state-transition in the scheme being simulated. Chander et al. [3] define the notions of strong and weak simulations. A strong simulation of A in B requires that B makes one state-transition when A makes one state-transition. A weak simulation requires that B makes a bounded (by a constant) number of state-transitions to simulate one state-transition in A. A main result in [3] is that a specific TM scheme considered there is more expressive than ACL because there exists no (strong or weak) simulation of the TM scheme in ACL. The proof is based on the observation that an unbounded (but still finite) number of state-transitions in ACL is required to simulate one state-transition in the TM scheme.

  On the other hand, an unbounded number of state-transitions is allowed by Sandhu and Ganta [19]. They use a simulation that involves an unbounded number of state-transitions to prove that ATAM (Augmented Typed Access Matrix) is equivalent in expressive power to TAM (Typed Access Matrix).

Although significant progress has been made in comparing access control models, this current state of art is unsatisfactory for the following reasons. First, different definitions of simulations make it impossible to put different results and claims about expressive power of access control models into a single context. For example, the result that RBAC is at least as expressive as DAC [15, 13] is qualitatively different from the result that TAM is at least as expressive as ATAM [19], as the former does not require the preservation of safety properties. These results are again qualitatively different from the result that ACL is less expressive than Trust Management [3], as the latter requires a bounded number of state-transitions in simulations.

Second, some definitions of simulations that are used in the literature are too weak to distinguish access control models from one another in a meaningful way. Sandhu et al. [13, 15, 17] show that various forms of DAC (including ATAM, in which simple safety is undecidable) can be simulated in RBAC, using the notion of simulations derived from the implementation paradigm. We show in this paper that using the same notion of simulations, RBAC can be simulated in strict DAC, one of the most basic forms of DAC where simple safety is trivially decidable. This suggests that using such a notion of simulations, it is likely that one can show that all access control models have the same expressive power. Thus, this notion of simulations is not very useful in differentiating between models based on expressive power.

Finally, the rationale for some choices made in existing definitions of simulations is often not clearly stated and justified. It is unclear why certain requirements are made or not made for simulations when comparing the expressive power of access control models. For instance, when a simulation involves an unbounded number of state-transitions, Ganta [4] considers this to be a "weak" simulation, while Chander et al. [3] do not consider this to be a simulation at all.

In this paper, we build on existing work and seek to construct uniform bases for comparing access control models. To determine the requirements on simulations in a systematic and justifiable manner, we start from the rationales and intuitions underlying different definitions for simulations. Our approach is to first identify the desirable and intuitive properties one would like simulations to have and then come up with the conditions on simulations that are both sufficient and necessary to satisfy those properties. Informally, what is desired is that when one scheme can represent all types of policies that another can, then the former is deemed to be at least as expressive as the latter. This observation is made by Ganta [4] as well.

Our theory is based on definitions of simulations that preserve security properties. Examples of such security properties are availability, mutual exclusion and bounded safety. Intuitively, such security properties are the sorts of policies one would want to represent in an access control system. *Security analysis* is used to verify that desired security properties are indeed maintained across state-transitions in an access control system. It was introduced by Li et al. [11], and generalizes the notion of safety analysis [6]. In this paper, we introduce compositional security analysis, which generalizes security analysis to consider logical combinations of queries in security analysis.

We introduce two notions of simulations called *state-matching reductions* and *reductions.* We show that state-matching reductions are necessary and sufficient for preserving compositional security properties and that reductions are necessary and sufficient for preserving security properties. A state-matching reduction reduces the compositional security analysis problem in one scheme to that in another scheme. A reduction reduces the security analysis problem in one scheme to that in another scheme.

To summarize, the contributions of this paper are as follows.

- We introduce a theory for comparing access control models based on the notions of state-matching reductions and reductions, together with detailed justifications for the design decisions.

- We analyze the deficiency of using the implementation paradigm to compare access control models and show that it leads to a weak notion of simulations and cannot be used to differentiate access control models from one another based on expressive power.

- We apply our theory in three cases. We show that:

  – there exists a reduction, but no state-matching reduction from Strict DAC with Change of Ownership (SDCO) to RBAC with ARBAC97 [16] as the administrative model. To our knowledge, this is the first evidence of the limitation of the expressive power of RBAC in comparison to DAC. RBAC has been compared to various forms of DAC, including SDCO, in the literature [15, 17].

  – there exists a state-matching reduction from RBAC with an administrative model that is a component of ARBAC97 [16] to RT [8, 9], a trust-management language.

  – there exists no state-matching reduction from ATAM to TAM. This solves an open problem stated by Sandhu and Ganta [19] by formalizing the benefit from the ability to check for the absence of rights in addition to the ability to check for the presence of rights.

The remainder of this paper is organized as follows. We present our theory for comparing access control models in Section 2. In Section 3, we analyze the implementation paradigm for simulations. In Section 4.1, we discuss comparisons of DAC to RBAC from the literature. In the remainder of Section 4, we apply our theory to compare the expressive power of schemes in three cases. We conclude with Section 5. Appendix A presents a "simulation" of RBAC in strict DAC, Appendix B gives precise characterizations of schemes used in Sections 4.2, 4.3 and 4.4. Proofs not included in the main body of the paper appear in Appendix C.

2 Comparisons Based on Security Analysis

A requirement used in the literature for simulations is the preservation of simple safety properties. Indeed, this is the only requirement on simulations in [1, 18, 19]. If a simulation of scheme A in scheme B satisfies this requirement, then a system in A reaches an unsafe state if and only if the system's mapping in B reaches an unsafe state. In other words, the result of simple safety analysis[1] is preserved by the simulation.

Simple safety analysis, i.e., determining whether an access control system can reach a state in which an unsafe access is allowed, was first formalized by Harrison et al. [6] in the context of the well-known access matrix model [5, 7]. In the HRU scheme [6], a protection system has a finite set of rights and a finite set of commands. A state of a protection system is an access control matrix, with rows corresponding to subjects, and columns corresponding to objects; each cell in the matrix is a set of rights. A command takes the form of "if the given conditions hold in the current state, execute a sequence of primitive operations." Each condition tests whether a right exists in a cell in the matrix. There are six kinds of primitive operations: enter a right into a specific cell in the matrix, delete a right from a cell in the matrix, create a new subject, create a new object, destroy an existing subject, and destroy an existing object. The following is an example command that allows the owner of a file to grant the read right to another user.

```
command grantRead(u1,u2,f)
 if  'own' in (u1,f)
 then enter 'read' into (u2,f)
end
```

In the example, `u1`, `u2` and `f` are formal parameters to the command. They are instantiated by objects (or subjects) when the command is executed. In [6], Harrison et al. prove that in the HRU scheme, the safety question is undecidable, by showing that any Turing machine can be simulated by a protection system.

Treating the preservation of simple safety properties as the sole requirement of simulations is based on the implicit assumption that simple safety is the *only* interesting property in access control schemes, an assumption that is not valid. When originally introduced in [6], simple safety was described as just one class of queries one can consider. Recently, Li et al. [11] introduced the notion of security analysis, which generalizes simple safety to other properties such as bounded safety, simple availability, mutual exclusion and containment.

In this section, we present a theory for comparing access control models based on the preservation of security properties.

2.1 Access Control Schemes and Security Analysis

Definition 1 (Access Control Schemes) An *access control scheme* is a state-transition system $\langle \Gamma, Q, \vdash, \Psi \rangle$, in which Γ is a set of states, Q is a set of queries, $\vdash : \Gamma \times Q \rightarrow \{true, false\}$ is called the entailment relation, and Ψ is a set of state-transition rules.

[1]What we call simple safety analysis is called safety analysis in the literature. In [11], more general notions of safety analysis, for which the traditional safety analysis is just a special case, were introduced. Here we follow the terminology in [11].

A *state*, $\gamma \in \Gamma$, contains all the information necessary for making access control decisions at a given time. The *entailment relation*, $\vdash$, determines whether a *query* is true or not in a given state. When a query, $q \in Q$, arises from an access request, $\gamma \vdash q$ means that the access request q is allowed in the state γ, and $\gamma \not\vdash q$ means that q is not allowed. Some access control schemes also allow queries other than those corresponding to a specific request, e.g., whether every subject that has access to a resource is an employee of the organization. Such queries can be useful for understanding the properties of complex access control systems.

A *state-transition rule*, $\psi \in \Psi$, determines how the access control system changes state. More precisely, ψ defines a binary relation (denoted by $\mapsto_\psi$) on Γ. Given $\gamma, \gamma_1 \in \Gamma$, we write $\gamma \mapsto_\psi \gamma_1$ if the change of state from γ to γ_1 is allowed by ψ, and $\gamma \overset{*}{\mapsto}_\psi \gamma_1$ if a sequence of zero or more allowed changes leads from γ to γ_1. In other words, $\overset{*}{\mapsto}_\psi$ is the transitive closure of $\mapsto_\psi$. If $\gamma \overset{*}{\mapsto}_\psi \gamma_1$, we say that γ_1 is *ψ-reachable* from γ, or simply γ_1 is *reachable*, when γ and ψ are clear from the context.

An *access control model* is a set of access control schemes. An *access control system* in an access control scheme $\langle \Gamma, Q, \vdash, \Psi \rangle$ is given by a pair (γ, ψ), where $\gamma \in \Gamma$ is the current state the system is in and $\psi \in \Psi$ the state-transition rule that governs the system's state changes.

Similar definitions for access control schemes appear in [1, 3]; our definition from above also appears in [10], and is different from the definitions in [1, 3] in the following two respects. First, our definition is more abstract in that it does not refer to subjects, objects, and rights and that the details of a state-transition rule are not specified. We find such an abstract definition more suitable to capture the notion of expressive power especially when the models or schemes that are compared are "structurally" different (e.g., a scheme based on RBAC that has a notion of roles that is an indirection between users and permissions, and a scheme based on the access-matrix model in which rights are assigned to subjects directly). Second, our definition makes the set of queries that can be asked an explicit part of the specification of an access control scheme. In existing definitions in the literature, the set of queries is often not explicitly specified. Sometimes, the implicit set of queries is clear from context; other times, it is not clear.

The HRU Scheme We now show an example access control scheme, the HRU scheme, that is derived from the work by Harrison et al. [6]. We assume the existence of three countably infinite sets: $\mathcal{S}$, $\mathcal{O}$, and $\mathcal{R}$, which are the sets of all possible subjects, objects, and rights. We further assume that $\mathcal{S} \subseteq \mathcal{O}$. In the HRU scheme:

- Γ is the set of all possible access matrices. Formally, each $\gamma \in \Gamma$ is identified by three sets, $S_\gamma \subset \mathcal{S}$, $O_\gamma \subset \mathcal{O}$, and $R_\gamma \subset \mathcal{R}$, and a function $M_\gamma[\,] : S_\gamma \times O_\gamma \to 2^{R_\gamma}$, where $M_\gamma[s, o]$ gives the set of rights that are in the cell.

- Q is the set of all queries having the form: $r \in [s, o]$, where $r \in \mathcal{R}$ is a right, $s \in \mathcal{S}$ is a subject, $o \in \mathcal{O}$ is an object. This query asks whether the right r exists in the cell corresponding to subject s and object o.

- The entailment relation is defined as follows: $\gamma \vdash r \in [s, o]$ if and only if $s \in S_\gamma$, $o \in O_\gamma$, and $r \in M_\gamma[s, o]$.

- Each state-transition rule ψ is given by a set of command schemas. Given ψ, the change from γ to γ_1 is allowed if there exists an instance of a command schema in ψ that when applied to γ gets γ_1.

The set of queries is not explicitly specified in [6]. It is conceivable to consider other classes of queries, e.g., comparing the set of all subjects that have a given right over a given object with another set of subjects. In our framework, HRU with different classes of queries can be viewed as different schemes in the access matrix model.

Definition 2 (Security Analysis) Given an access control system $\langle \Gamma, Q, \vdash, \Psi \rangle$, a *security analysis instance* has the form $\langle \gamma, q, \psi, \Pi \rangle$, where $\gamma \in \Gamma$ is a state, $q \in Q$ is a query, $\psi \in \Psi$ is a state-transition rule, and $\Pi \in \{\exists, \forall\}$ is a quantifier.

An instance $\langle \gamma, q, \psi, \exists \rangle$ is said to be *existential*; it asks whether there exists γ_1 such that $\gamma \overset{*}{\mapsto}_\psi \gamma_1$ and $\gamma_1 \vdash q$? If so, we say q is *possible* (given γ and ψ).

An instance $\langle \gamma, q, \psi, \forall \rangle$ is said to be *universal*; it asks whether for every γ_1 such that $\gamma \overset{*}{\mapsto}_\psi \gamma_1$, $\gamma_1 \vdash q$? If so, we say q is *necessary* (given γ and ψ).

Simple safety analysis is a special case of security analysis. A simple safety analysis instance that asks whether a system (γ, ψ) in the HRU scheme can reach a state in which the subject s has the right r over the object o is represented as the following instance: $\langle \gamma, r \in [s, o], \psi, \exists \rangle$. The universal version of this instance, $\langle \gamma, r \in [s, o], \psi, \forall \rangle$, asks whether s always has the right r over the object o in every reachable state. Thus it refers to the availability property and asks whether a particular access right is always available to the subject s.

We now introduce a generalized notion of security analysis.

Definition 3 (Compositional Security Analysis) Given a scheme $\langle \Gamma, Q, \vdash, \Psi \rangle$, a *compositional security analysis* instance has the form $\langle \gamma, \phi, \psi, \Pi \rangle$, where γ, ψ, and Π are the same as in a security analysis instance, and ϕ is a propositional formula over Q, i.e., ϕ is constructed from queries in Q using propositional logic connectives such as $\wedge$, $\vee$, $\neg$.

For example, the compositional security analysis instance$\langle \gamma, (r_1 \in [s, o_1]) \wedge (r_2 \in [s, o_2]), \psi, \exists \rangle$ asks whether the system (γ, ψ) can reach a state in which s has both the right r_1 over o_1 and the right r_2 over o_2. Whether we should use security analysis or compositional security analysis is related to what types of policies we want to represent, and what types of policies we want to use as bases to compare the expressive power of different access control models or schemes. With compositional security analysis, we would be comparing models or schemes based on types of policies that are broader than with security analysis. For instance, if our set of queries Q contains queries related to users' access to files, then with compositional security analysis we can consider policies such as "Bob should never have write access to a particular file so long as his wife, Alice has a user account (and thus has some type of access to some file)."

2.2 Two Types of Reductions

In this section, we introduce the notions of reductions and state-matching reductions that we believe are adequate for comparing the expressive power of access control models. Before we introduce reductions, we discuss two types of mappings between access control schemes.

Definition 4 (Mapping) Given two access control schemes $A = \langle \Gamma^A, Q^A, \vdash^A, \Psi^A \rangle$ and $B = \langle \Gamma^B, Q^B, \vdash^B, \Psi^B \rangle$. A *mapping* from A to B is a function σ that maps each pair $\langle \gamma^A, \psi^A \rangle$ in A to a pair $\langle \gamma^B, \psi^B \rangle$ in B and maps each query q^A in A to a query q^B in B. Formally, $\sigma : (\Gamma^A \times \Psi^A) \cup Q^A \to (\Gamma^B \times \Psi^B) \cup Q^B$.

Definition 5 (Security-Preserving Mapping) A mapping σ is said to be *security-preserving* when every security analysis instance in A is true if and only if the *image* of the instance is true. Given a mapping $\sigma : (\Gamma^A \times \Psi^A) \cup Q^A \to (\Gamma^B \times \Psi^B) \cup Q^B$, the *image* of a security analysis instance $\langle \gamma^A, q^A, \psi^A, \Pi \rangle$ *under* σ is $\langle \gamma^B, q^B, \psi^B, \Pi \rangle$, where $\langle \gamma^B, \psi^B \rangle = \sigma(\langle \gamma^A, \psi^A \rangle$ and $q^B = \sigma(q^A)$.

The notion of security-preserving mappings captures the intuition that simulations should preserve security properties. Given a security-preserving mapping from A to B and an algorithm for solving the security analysis problem in B, one can construct an algorithm for solving the security analysis problem in A using the mapping. Also, security analysis in B is at least as hard as security analysis in A, modulo the efficiency of the mapping.

If an efficient (polynomial-time) mapping from A to B exists, and security analysis in A is intractable (or undecidable), then security analysis in B is also intractable (undecidable). Security preserving mappings are not powerful enough for comparisons of access control schemes based on compositional security analysis. We need the notion of a strongly security-preserving mapping for that purpose.

Definition 6 (Strongly Security-Preserving Mapping) Given a mapping σ from scheme A to scheme B, the image of a compositional analysis instance, $\langle \gamma^A, \phi^A, \psi^A, \Pi \rangle$, in A is $\langle \gamma^B, \phi^B, \psi^B, \Pi \rangle$, where $\langle \gamma^B, \psi^B \rangle = \sigma(\langle \gamma^A, \psi^A \rangle)$ and ϕ^B is obtained by replacing every query q^A in ϕ^A with $\sigma(q^A)$ (we abuse the terminology slightly and write $\phi^B = \sigma(\phi^A)$). A mapping σ from A to B is said to be *strongly security-preserving* when every compositional security analysis instance in A is true if and only if the image of the instance is true.

While the notions of security-preserving mappings capture the intuition that simulations should preserve security properties, they are not convenient for us to use directly. Using the definition for either type of mapping to directly prove that the mapping is (strongly) security preserving involves performing security analysis, which is expensive. We now introduce the notions of reductions, which state structural requirements on mappings for them to be security preserving. We start with a form of reduction appropriate for compositional security analysis and then discuss weaker forms.

Definition 7 (State-Matching Reduction) Given a mapping from A to B, $\sigma : (\Gamma^A \times \Psi^A) \cup Q^A \rightarrow (\Gamma^B \times \Psi^B) \cup Q^B$, we say that the two states γ^A and γ^B are *equivalent* under the mapping σ when for every $q^A \in Q^A$, $\gamma^A \vdash^A q^A$ if and only if $\gamma^B \vdash^B \sigma(q^A)$. A mapping σ from A to B is said to be a *state-matching reduction* if for every $\gamma^A \in \Gamma^A$ and every $\psi^A \in \Psi^A$, $\langle \gamma^B, \psi^B \rangle = \sigma(\langle \gamma^A, \psi^A \rangle)$ has the following two properties:

1. For every state γ_1^A in scheme A such that $\gamma^A \overset{*}{\mapsto}_{\psi} \gamma_1^A$, there exists a state γ_1^B such that $\gamma^B \overset{*}{\mapsto}_{\psi^B} \gamma_1^B$ and γ_1^A and γ_1^B are equivalent under σ.
2. For every state γ_1^B in scheme B such that $\gamma^B \overset{*}{\mapsto}_{\psi^B} \gamma_1^B$, there exists a state γ_1^A such that $\gamma^A \overset{*}{\mapsto}_{\psi} \gamma_1^A$ and γ_1^A and γ_1^B are equivalent under σ.

Property 1 says that for every state γ_1^A that is reachable from γ^A, there exists a reachable state in scheme B that is equivalent, i.e., answers all queries in the same way. Property 2 says the reverse, for every reachable state in B, there exists an equivalent state in A. The goal of these two properties is to guarantee that compositional security analysis results are preserved across the mapping. With the following theorem, we justify Definition 7.

Theorem 1 *Given two schemes A and B, a mapping σ from A to B is strongly security-preserving if and only if σ is a state-matching reduction.*

Proof.

The "if" direction. When σ is a state-matching reduction, given a compositional security analysis instance $\langle \gamma^A, \phi^A, \psi^A, \Pi \rangle$ in scheme A, let $\langle \gamma^B, \psi^B \rangle = \sigma(\langle \gamma^A, \psi^A \rangle)$ and $\phi^B = \sigma(\phi^A)$, we show that $\langle \gamma^A, \phi^A, \psi^A, \Pi \rangle$ is true if and only if $\langle \gamma^B, \phi^B, \psi^B, \Pi \rangle$ is true.

First consider the case that the instance $\langle \gamma^A, q^A, \psi^A, \Pi \rangle$ is existential, i.e., Π is $\exists$. If the instance is true, i.e., there exists a reachable state γ_1^A in which ϕ^A is true. Property 1 in Definition 7 guarantees that there exists a reachable state γ_1^B that is equivalent to γ_1^A; thus ϕ^B is true in γ_1^B; therefore, the instance in B, $\langle \gamma^B, \phi^B, \psi^B, \exists \rangle$, is also true. On the other hand, if $\langle \gamma^B, \phi^B, \psi^B, \exists \rangle$ is true, then there exists a reachable state γ_1^B in which ϕ^B is true. Property 2 in Definition 7 guarantees that there exists a state in A in which the analysis instance in A is true.

Now consider the case that the instance $\langle \gamma^A, \phi^A, \psi^A, \Pi \rangle$ is universal, i.e., Π is $\forall$. If the instance is false, i.e., there exists a reachable state γ_1^A in which ϕ^A is false. Property 1 guarantees that the instance in B is also false. Similarly, if the instance in B is false, then the instance in A is also false.

The "only if" direction. When σ is not a state-matching reduction, then there exists $\gamma^A \in \Gamma^A$ and $\psi^A \in \Psi^A$ such that $\langle \gamma^B, \psi^B \rangle = \sigma(\langle \gamma^A, \psi^A \rangle)$ violates one of the two properties in Definition 7.

First consider the case that Property 1 is violated. There exists a reachable state γ_1^A such that no state reachable from γ^B is equivalent to γ_1^A. Construct a formula ϕ^A as follows: ϕ^A is a conjunction of queries in Q or their complement. For every query query q^A in Q^A, ϕ^A includes q^A if $\gamma_1^A \vdash^A q^A$ and $\neg q^A$ if $\gamma_1^A \vdash^A q^A$. (Note that the length of ϕ^A may be infinite, since the total number of queries may be infinite.) Clearly, ϕ^A is true in γ_1^A, but $\sigma(\phi^A)$ is false in all states reachable from γ^B. Thus, the existential compositional analysis instance involving ϕ^A has different answers, and σ is not strongly security preserving.

Then consider the case that Property 2 is violated. There exists a state γ_1^B reachable from γ^B such that no state reachable from γ^A is equivalent to γ_1^B. Construct a formula ϕ^A as follows: ϕ^A is a conjunction of queries in Q or their complement. For every query query q^A in Q^A, ϕ^A includes q^A if $\gamma_1^B \vdash^B \sigma(q^A)$ and $\neg q^A$ if $\gamma_1^B \vdash^B \sigma(q^A)$. Clearly, ϕ^A is false in in all states reachable from γ^A, but $\sigma(\phi^A)$ is true in γ_1^B; thus, the existential compositional analysis instance involving ϕ^A has different answers, and σ is not strongly security preserving. ∎

Note that the proof uses a compositional analysis instance that contains a potentially infinite-length formula. If one chooses to restrict the formulas in analysis instances to be finite length, then state-matching reduction may not be necessary for being strongly security-preserving. Also, a state-matching reduction preserves compositional security properties. If we only need queries from Q to represent our policies and not compositions of those queries, then the following weaker notion of reductions is more suitable. However, we believe that the notion of state-matching reductions is quite natural by itself, and certainly necessary when compositional queries are of interest.

Definition 8 (Reduction) Given two access control schemes $A = \langle \Gamma^A, Q^A, \vdash^A, \Psi^A \rangle$ and $B = \langle \Gamma^B, Q^B, \vdash^B, \Psi^B \rangle$. A mapping from A to B, σ, is said to be a *reduction* from A to B if for every $\gamma^A \in \Gamma^A$ and every $\psi^A \in \Psi^A$, $\langle \gamma^B, \psi^B \rangle = \sigma(\langle \gamma^A, \psi^A \rangle)$ has the following two properties:

1. For every state γ_1^A and every query q^A in scheme A, if $\gamma^A \stackrel{*}{\mapsto}_{\psi} \gamma_1^A$, then in scheme B there exists a state γ_1^B such that $\gamma^B \stackrel{*}{\mapsto}_{\psi^B} \gamma_1^B$ and $\gamma_1^A \vdash^A q^A$ if and only if $\gamma_1^B \vdash^B \sigma(q^A)$.

2. For every state γ_1^B in scheme B and every query q^A in scheme A, if $\gamma^B \stackrel{*}{\mapsto}_{\psi^B} \gamma_1^B$, there exists a state γ_1^A such that $\gamma^A \stackrel{*}{\mapsto}_{\psi} \gamma_1^A$ and $\gamma_1^A \vdash^A q^A$ if and only if $\gamma_1^B \vdash^B \sigma(q^A)$.

Definition 7 differs from Definition 8 in that the former requires that for every reachable state in A (B, resp.) there exist a matching state in B (A, resp.) that gives the same answer for *every query*. Definition 8 requires the existence of a matching state for every query; however, the matching states may be different for different queries. Property 1 in Definition 8 says that for every reachable state in A and every query in A, there exists a reachable state in B that gives the same answer to (the image of) the query. Property 2 says the reverse direction. The goal of these two properties is to guarantee that security analysis results are preserved across the mapping. The fact that a reduction, as defined in Definition 8, is adequate for preserving security analysis results is formally captured by the following theorem.

Theorem 2 *Given two schemes A and B, a mapping, σ, from A to B is security preserving if and only if σ is a reduction.*

Proof. **The "if" direction.** When σ is a reduction, given a security analysis instance $\langle \gamma^A, q^A, \psi^A, \Pi \rangle$ in scheme A, let $\langle \gamma^B, \psi^B \rangle = \sigma(\langle \gamma^A, \psi^A \rangle)$ and $q^B = \sigma(q^A)$, we show that $\langle \gamma^A, q^A, \psi^A, \Pi \rangle$ is true if and only if $\langle \gamma^B, q^B, \psi^B, \Pi \rangle$ is true.

First consider the case that the instance $\langle\gamma^A, q^A, \psi^A, \Pi\rangle$ is existential, i.e., Π is $\exists$. If the instance is true, i.e., there exists a reachable state γ_1^A in which q^A is true. Property 1 in Definition 8 guarantees that there exists a reachable state γ_1^B in which q^B is true. Therefore, the instance in B, $\langle\gamma^B, q^B, \psi^B, \exists\rangle$, is also true. On the other hand, if $\langle\gamma^B, q^B, \psi^B, \exists\rangle$ is true, then there exists a reachable state γ_1^B in which q^B is true. Property 2 in Definition 8 guarantees that there exists a state in A in which q^A is true; thus the analysis instance in A is true.

Now consider the case that the instance $\langle\gamma^A, q^A, \psi^A, \Pi\rangle$ is universal, i.e., Π is $\forall$. If the instance is false, i.e., there exists a reachable state γ_1^A in which q^A is false. Property 1 guarantees that the instance in B is also false. Similarly, if the instance in B is false, then the instance in A is also false.

The "only if" direction. When σ is not a reduction, then there exists $\gamma^A \in \Gamma^A$ and $\psi^A \in \Psi^A$ such that $\langle\gamma^B, \psi^B\rangle = \sigma(\langle\gamma^A, \psi^A\rangle)$ violates one of the two properties in Definition 8.

First consider the case that Property 1 is violated. There exists a reachable state γ_1^A and a query q^A such that for every state reachable from γ^B the answer for the query $\sigma(q^A)$ under the state is different from the answer for q^A under γ_1^A. If $\gamma_1^A \vdash^A q^A$, then this means that q^B is false in every state reachable from γ^B. Thus the security analysis instance $\langle\gamma^A, q^A, \psi^A, \exists\rangle$ is true, but its image under σ is false. Thus, the mapping σ is not security-preserving. If $\gamma_1^A \not\vdash^A q^A$, then this means that q^B is true in every state reachable from γ^B. Thus the security analysis instance $\langle\gamma^A, q^A, \psi^A, \forall\rangle$ is false, but its image under σ is true.

Then consider the case that Property 2 is violated. There exists a state γ_1^B reachable from γ^B and a query q^A such that for every state reachable from γ^A the answer for the query q^A under the state is different from the answer for $\sigma(q^A)$ under γ_1^B. If $\gamma_1^B \vdash^B \sigma(q^A)$, then this means that q^A is false in every state reachable from γ^A. Thus the security analysis instance $\langle\gamma^A, q^A, \psi^A, \exists\rangle$ is false, but its image under σ is true. If $\gamma_1^B \not\vdash^B q^B$, then this means that q^A is true in every state reachable from γ^A. Thus the security analysis instance $\langle\gamma^A, q^A, \psi^A, \forall\rangle$ is true, but its mapping in B is false. ∎

Comparisons of two access control models are based on comparisons among access control schemes based on those models. Comparisons of two access control schemes, in turn, are based on whether only the queries from Q need to be represented, or compositions of those queries need to be represented as well.

Definition 9 (Comparing the Expressive Power of Access Control Models) Given two access control models $\mathcal{M}$ and $\mathcal{M}'$, we say that $\mathcal{M}'$ is at least as expressive as $\mathcal{M}$ (or $\mathcal{M}'$ has at least as much expressive power as $\mathcal{M}'$) if for every scheme in $\mathcal{M}$ there exists a state-matching reduction (or a reduction) from it to a scheme in $\mathcal{M}'$. In addition, if for every scheme in $\mathcal{M}'$, there exists a state-matching reduction (reduction) from it to a scheme in $\mathcal{M}$, then we say that M and M' are equivalent in expressive power. If $\mathcal{M}'$ is at least as expressive as the $\mathcal{M}$, and there exists a scheme A in $\mathcal{M}'$ such that for any scheme B in $\mathcal{M}$, no state-matching reduction (reduction) from A to B exists, we say that $\mathcal{M}'$ is strictly more expressive than $\mathcal{M}$.

We compare the expressive power of two schemes based on state-matching reductions when compositional queries are needed to represent the policies of interest. Otherwise, reductions suffice. Observe that we can use the above definition to compare the expressive power of two access control schemes A and B, by viewing each scheme as an access control model consists of just that scheme.

2.3 Discussions of alterative definitions for reduction

In this section, we discuss alternative definitions that differ slightly from the ones discussed in the previous section. The first of these definitions is used by Sandhu and Ganta [18, 19] for simulations.

Definition 10 (Form-1 Weak Reduction) A mapping from A to B, given by $\sigma : (\Gamma^A \times \Psi^A) \cup Q^A \rightarrow (\Gamma^B \times \Psi^B) \cup Q^B$, is a *form-1 weak reduction* if for every $\gamma^A \in \Gamma^A$ and every $\psi^A \in \Psi^A$, $\langle\gamma^B, \psi^B\rangle = \sigma(\langle\gamma^A, \psi^A\rangle)$ has the following two properties:

1. For every query q^A, if there exists a state γ_1^A in scheme A such that $\gamma^A \overset{*}{\mapsto}_{\psi^A} \gamma_1^A$ and $\gamma_1^A \vdash^A q^A$, then there exists a state γ_1^B such that $\gamma^B \overset{*}{\mapsto}_{\psi^B} \gamma_1^B$ and $\gamma_1^B \vdash^B \sigma(q^A)$.

2. For every query q^A, if there exists γ_1^B in scheme B such that $\gamma^B \overset{*}{\mapsto}_{\psi^B} \gamma_1^B$ and $\gamma_1^B \vdash^B \sigma(q^A)$, then there exists a state γ_1^A such that $\gamma^A \overset{*}{\mapsto}_{\psi} \gamma_1^A$ and $\gamma_1^A \vdash^A q^A$ if and only if $\gamma_1^B \vdash^B \sigma(q^A)$.

The intuition underlying Definition 10, as stated by Sandhu [18] is, "systems are equivalent if they have equivalent worst case behavior". Therefore, simulations only need to preserve the worst-case access. Definition 10 is weaker than Definition 8 in that it requires the existence of a matching state when a query is true in the state, but does not require so when the query is false. Therefore, it is possible that a query q^A is true in all states that are reachable from γ^A, but the query $\sigma(q^A)$ is false in some states that are reachable from γ^B (the query $\sigma(q^A)$ needs to be true in at least one state reachable from γ^B). This indicates that Definition 10 does not preserve answers to universal security analysis instances. Definition 10 is adequate for the purposes in [18, 19] since only simple safety analysis (which is existential) was considered there.

The decision of defining a mapping to be a function from $(\Gamma^A \times \Psi^A) \cup Q^A$ to $(\Gamma^B \times \Psi^B) \cup Q^B$ also warrants some discussion. One alternative is to define a mapping from A to B to be a function that maps each state in A to a state in B, each state-transition rule in A to a state-transition rule in B, and each query in A to a query in B. Such a function would be denoted as $\sigma : \Gamma^A \cup \Psi^A \cup Q^A \to \Gamma^B \cup \Psi^B \cup Q^B$. One can verify any such function is also a mapping according to Definition 4, which gives more flexibility in terms of mapping states and state-transition rules from A to B. By Definition 4, the state corresponding to a state γ^A may also depends upon the state-transition being considered.

Another alternative is to define a mapping from A to B to be a function $\sigma : \Gamma^A \times \Psi^A \times Q^A \to \Gamma^B \times \Psi^B \times Q^B$, in other words, the mapping of states, state-transition rules, and queries may depend on each other. This definition will also leads to a weaker notion of reduction:

Definition 11 (Form-2 Weak Reduction) A form-2 weak reduction from A to B is a function $\sigma : \Gamma^A \times \Psi^A \times Q^A \to \Gamma^B \times \Psi^B \times Q^B$ such that for every $\gamma^A \in \Gamma^A$, every $\psi^A \in \Psi^A$, and every $q^A \in Q^A$, $\langle \gamma^B, \psi^B, q^B \rangle = \sigma(\langle \gamma^A, \psi^A, q^A \rangle)$ has the following two properties:

1. For every state γ_1^A in scheme A such that $\gamma^A \overset{*}{\mapsto}_{\psi} \gamma_1^A$, there exists a state γ_1^B such that $\gamma^B \overset{*}{\mapsto}_{\psi^B} \gamma_1^B$ and $\gamma_1^A \vdash^A q^A$ if and only if $\gamma_1^B \vdash^B q^B$.

2. For every state γ_1^B in scheme B such that $\gamma^B \overset{*}{\mapsto}_{\psi^B} \gamma_1^B$, there exists a state γ_1^A such that $\gamma^A \overset{*}{\mapsto}_{\psi} \gamma_1^A$ and $\gamma_1^A \vdash^A q^A$ if and only if $\gamma_1^B \vdash^B q^B$.

It is not difficult to prove that a Form-2 weak reduction is also security preserving, in the sense that any security analysis instance $\langle \gamma^A, q^A, \psi^A, \Pi \rangle$ in A can be mapped to a security analysis in B. However, it is not a mapping, as the mapping of states and state-transition rules may depend on the query.

Definition 11 is used implicitly in Theorems 2 and 3 in [10] for reductions from two RBAC schemes to the RT Role-based Trust-management framework [9, 11]. As we state in Theorem 5 in this paper, a reduction used there for one of the RBAC schemes can be changed to a security-preserving mapping in a straightforward manner.

We choose not to adopt this weaker notion of reduction for the following reason. Under this definition, given an access control system (γ^A, ψ^A), to answer n analysis instances involving different queries, one has to do n translations of states and state-transitions, which are often time consuming. While using Definition 4 and Definition 8, one can do the mapping of (γ^A, ψ^A) once and use it to answer all n analysis instances.

A third weak form of reduction is introduced by Ammann et al. [1]. That work discusses the expressive power of multi-parent creation when compared to single-parent creation.

Definition 12 (Form-3 Weak Reduction) A mapping from A to B, given by $\sigma : (\Gamma^A \times \Psi^A) \cup Q^A \to (\Gamma^B \times \Psi^B) \cup Q^B$, is a *form-3 weak reduction* if for every $\gamma^A \in \Gamma^A$ and every $\psi^A \in \Psi^A$, $\langle \gamma^B, \psi^B \rangle = \sigma(\langle \gamma^A, \psi^A \rangle)$ has the following two properties:

1. For every state γ_1^A and every query q^A in scheme A, if $\gamma^A \overset{*}{\mapsto}_\psi \gamma_1^A$, then in scheme B there exists a state γ_1^B such that $\gamma^B \overset{*}{\mapsto}_{\psi^B} \gamma_1^B$ and $\gamma_1^A \vdash^A q^A$ if and only if $\gamma_1^B \vdash^B \sigma(q^A)$.

2. For every state γ_1^B in scheme B and every query q^A in scheme A, if $\gamma^B \overset{*}{\mapsto}_{\psi^B} \gamma_1^B$, then either (a) there exists a state γ_1^A such that $\gamma^A \overset{*}{\mapsto}_\psi \gamma_1^A$ and $\gamma_1^A \vdash^A q^A$ if and only if $\gamma_1^B \vdash^B \sigma(q^A)$, or (b) there exists a state γ_2^B such that $\gamma_1^B \overset{*}{\mapsto}_{\psi^B} \gamma_2^B$ and a state γ_1^A such that $\gamma^A \overset{*}{\mapsto}_\psi \gamma_1^A$, and $\gamma_1^A \vdash^A q^A$ if and only if $\gamma_2^B \vdash^B \sigma(q^A)$.

As pointed out by Ammann et al. [1], this form of reduction suffices for monotonic schemes — those schemes in which once a state is reached in which a query is true, in all reachable states from that state, the query remains true. Therefore, this form of reduction cannot be used to compare schemes when queries can become false after being true. As with the reduction from Definition 10, this form of reduction cannot be used for universal queries.

3 The Implementation Paradigm for Simulation: An Examination

Several authors use the implementation paradigm for simulations, e.g., Osborn et al. [15] state that “a positive answer [to the question whether LBAC (lattice-based access control) can be simulated in RBAC] is also practically significant, because it implies that the same Trust Computing Base can be configured to enforce RBAC in general and LBAC in particular.” However, in these papers [13, 15, 17], a precise definition for simulations is not given. This makes the significance of such results unclear, at least in terms of comparing the expressive power of different access control models.

In this section, we analyze the implementation paradigm and argue that this does not lead to a notion of simulations that is meaningful for comparing the expressive power of different access control models. More precisely, the notions of simulations derived from this paradigm are so weak that almost all access control schemes are equivalent.

To formalize the implementation paradigm for simulation, a natural goal is to use an implementation of an access control scheme for another scheme. Intuitively, if a scheme A can be simulated in a scheme B, then there exists a *simulator* that, when given access to an interface to (an implementation of) B, can provide an interface that is exactly the same as the interface to (an implementation of) A.

When considering the interface of an access control scheme, we have to consider how state-transitions occur. Intuitively, an access control system changes its state because some actors (subjects, principals, users, etc.) initiate certain actions. An implementation of an access control scheme thus has an interface consisting of at least the following functions:

- $init(\gamma)$: set the current state to γ.
- $query(q)$: ask the query q and receives a yes/no response.
- $apply(a)$: apply the action a on the system, which may result in a state-transition in the system.
- functions providing other capabilities, e.g., traversing the subjects and objects in the system.

A simulator of A in B is thus a program that takes an interface of B and provides an interface of A that is indistinguishable from an implementation for A. In other words, the simulator is a blackbox that when given access to a backbox implementation of B, gives an implementation of A. This intuition seems to make sense if the goal is to use an implementation of B to implement A.

It is tempting to start formalizing the above intuition; however, there are several subtle issues that need to be resolved first.

As can be easily seen, for any two schemes A and B, a trivial simulator exists. The simulator implements all the functionalities of A by itself, without interacting with the implementation of B. Clearly, one would like to rule out these trivial simulators. One natural way to do so is to restrict the amount of space used by the simulator to be sub-linear in the size of the state of the scheme it is simulating. It *seems* to be a reasonable requirement that the simulator takes constant space on its own, i.e., the space used by the simulator does not depend on the size of the state. (The space used by the implementation of B is not considered here.)

Another issue is whether to further restrict a simulator's internal behavior. When the simulator receives a query in the scheme A, it may issue multiple queries to the blackbox implementation of B before answering the query; it may even perform some state-transition on B before answering the query. Similarly, the simulator may perform multiple queries and state-transitions on B to simulate one state-transition in A.

If no restriction is placed, then the notion of simulation is too weak to separate different access control models. For example, in [13], Munawer and Sandhu constructed a simulation of ATAM in RBAC. In Appendix A, we give a simulation of RBAC in strict DAC, a discretionary model that allows only the owner of an object to grant rights over the object to another subject and ownership cannot be transferred. According to these results, the simplest DAC (in which security analysis is efficiently decidable) has the same expressive power as ATAM (in which simple safety analysis is undecidable). This illustrates the point that, without precise requirements, simulation is not a very useful concept for comparing access control models.

If one places restrictions on the simulator, then the question is what restrictions are reasonable. Our conclusion is that it is very difficult to justify such requirements. In the following, we elaborate on this.

One possibility is to restrict the internal behavior of the simulator, e.g., to restrict it to issue only one query to B in order to answer one query in A and to make bounded number of state-transitions in B to simulate one state-transition in A. Under these restrictions, one can prove that RBAC cannot be simulated in the HRU model. The assignment of a user to a role in RBAC results in the user gaining all the accesses to objects implied by the permissions associated with that role; therefore, it changes the answers to an unbounded number of queries (queries involving those permissions.) One may argue that the assignment of a user to a role is a single "action" in RBAC, and therefore, the acquiring of those permissions by that user is accomplished in a single "action." The corresponding assignment of rights in the HRU access matrix cannot be accomplished by a single command, or a bounded number of command for that matter, as each command only changes a bounded number of cells in the matrix. Thus, any mapping of the user-assignment in RBAC involves an unbounded number of commands being executed in HRU. Nonetheless, one can argue that this is balanced by the efficiency of checking whether a user has a particular right in the two models. A naive implementation of an RBAC model may involve having to collect all roles to which that user is assigned, and then collecting all permissions associated with those roles, and then checking whether one of those permissions corresponds to the object and access right for which we are checking. The time this process takes depends on the size of the current state and is unbounded. The corresponding check in HRU is simpler: we simply check whether the corresponding access right exists in the cell in the matrix. Thus, we can argue that there is a trade-off between time-to-update, and time-to-check-access between the two schemes.

Another possibility is to measure how much time the simulator takes to perform a state-transition and to answer one query in the worst case and require that there cannot be a significant slowdown. This possibility is complicated by the fact that the efficiency of these operations are not predetermined in any access control scheme, the implementation can make trade-offs between time complexity and space complexity and between query answering and state-transitions. Any comparison must involve at least three axes, query time, state-transition time, and space. Furthermore, the best ways to implement an access control scheme is not always known. Finally, these implementation-level details do not seem to belong in the comparison of access control models; as such models by themselves are abstract models to study properties other than efficiency.

In summary, our analysis in this section suggests that the “implementation paradigm” does not seem to yield effective definitions of simulations that are useful to compare access control models. This also suggests that expressive power results proved under this paradigm should be reexamined.

An alternate approach Bertino et al. [2] propose a different implementation paradigm from the one discussed above. They present a framework based on logic programming within which to compare the expressive power of access control models. A library of logic facts and rules are provided, and each access control model is a collection of some facts and rules from that library. Access control models are then compared based on what facts and rules are used to represent each of them. The approach in that work is structural: if in one model we use certain facts and rules, but not in another, then the two models are incomparable. Furthermore, if one model uses more facts and rules than another, then the former is more expressive than the latter. This basis is used in arguing that RBAC is more expressive than MAC as RBAC has the notion of roles. State-transitions are not considered in this approach, and the preservation of properties across state-transitions is not part of the bases for comparison. Our theory for comparing the expressive power of access control models is based on whether schemes from one model can represent policies that schemes from another cannot. We do not have any structural restrictions in comparing two models. Thereby, our work is fundamentally different from the work by Bertino et al. [2].

4 Applying the Theory

In this section, we apply our theory from Section 2 to compare the expressive power of different access control schemes. We examine two particular results from literature using our theory: (1) that RBAC is at least as expressive as DAC (Sections 4.1 and 4.2), and (2) that TAM is at least as expressive as ATAM (Section 4.4). We show also that the trust management language $\mathsf{RT}[\cap]$ is at least as expressive as an RBAC scheme (Section 4.3). Our schemes are precisely characterized in Appendix B, and proofs for our results are in Appendix C.

4.1 Examining comparisons of RBAC and DAC

Munawer and Sandhu [13] present a simulation of ATAM in RBAC and conclude that RBAC is at least as expressive as ATAM. Osborn et al. [15, 14, 17] give simulations of various MAC and DAC schemes in RBAC. The main conclusion of Osborn et al. [15, 14, 17] is that as MAC and DAC can be simulated in RBAC, a Trusted Computing Based (TCB) needs to include an implementation of RBAC only, and DAC and MAC policies can be successfully represented and enforced by the TCB.

In the simulations used in [13, 15, 14, 17], the preservation of safety (or other security) properties is not identified as an objective. From the above conclusion in [15, 14, 17], it seems that they follow the implementation paradigm. As discussed in Section 3, this paradigm leads to a weak notion of simulations, as exemplified by the simulation of RBAC in strict DAC in Appendix A.

We observe also that the problem of comparing RBAC with DAC as stated by Osborn et al. [15, 17] is ill-defined (or at least not clearly defined). RBAC by itself only specifies the structures to store access control information, but not how to manipulate these structures, which are specified by administrative models. In other words, only the set Γ of states is precisely defined, the set Ψ of state-transition rules is not. The counterpart of RBAC is the access matrix model, instead of DAC (or MAC). In DAC, we specify that access control information is stored in a matrix, and we also specify rules on how to change the access matrix. The statement that RBAC is at least as expressive as DAC (or MAC) is similar to saying that the access matrix model is at least as expressive as DAC or MAC. Comparing the RBAC model with the access matrix model is not fruitful either, as both models can include arbitrary state-transition rules.

4.2 Comparing ARBAC97 with a form of DAC

To compare any RBAC-based model with DAC, one needs to specify the administrative model (state-transition rules) for RBAC. In existing comparisons of RBAC and DAC [13, 15, 17], new and rather complicated administrative models are introduced "on the fly" to simulate the effects in DAC. In this section, we compare the expressive power of RBAC with ARBAC97 [16] as the administrative model to that of SDCO, a rather simple form of DAC. A precise characterization of SDCO is in Appendix B.1, and of the ARBAC97 scheme is in Appendix B.2. Osborn et al. [15] assert that SDCO can be simulated in RBAC. We assert that there does not exist a state-matching reduction from SDCO to the ARBAC97 scheme, given a natural query set for each scheme.

This result is significant as it shows that we cannot assert that RBAC is more expressive than DAC without qualifying the assertion; a strongly security-preserving mapping does not exist from SDCO to ARBAC97. Our conclusion provides the first evidence that the expressive power of RBAC (or at least some reasonable incarnation of it) is limited.

Theorem 3 *There exists a reduction from SDCO to ARBAC97.*

Theorem 4 *There exists no state-matching reduction from SDCO to ARBAC97.*

The proofs are in Appendices C.1 and C.2 respectively. One may ask whether there are other schemes based on RBAC for which there is indeed a state-matching reduction from SDCO. An approach may be to adopt a different query set for ARBAC97. We observe that for certain other query sets as well, the non-existence of a state-matching reduction holds. As an example, suppose we map the query for the presence of a right in SDCO to a query for the absence of a permission in RBAC. In this case as well, there exists no state-matching reduction from SDCO. Whether there exists a meaningful set of state-transition rules (an administrative model) for RBAC for which there is a state-matching reduction from SDCO is an open problem.

4.3 Comparing an RBAC scheme with a Trust Management Language

In this section, we compare a particular RBAC scheme to the trust management language, $\mathsf{RT}[\cap]$. The RBAC scheme we consider is called Assignment And Revocation (AAR) [10]. In AAR, the state is an RBAC state, and state-transition rules are those from the URA97 component of the ARBAC97 [16]; users may be assigned to and revoked from roles. Precise characterizations of AAR are in [10] and Appendix B.3.

$\mathsf{RT}[\cap]$ is a trust management language in which a state is a set of credentials issued by the principals involved in the system. A credential denotes membership in a principal's role. A credential is one of three types: (1) A principal is asserted to be a member of another principal's role, (2) All the principals that are members of a principal's role are asserted to also be members of another principal's role, and (3) All the principals that are members of two roles (the intersection of the members of the roles) are also members of another principal's role. Some details on $\mathsf{RT}[\cap]$ are in Appendix B.4, and we refer the reader to Li et al. [9, 11, 12] for more details on $\mathsf{RT}[\cap]$.

Li and Tripunitara [10] present a form-2 weak reduction (see Definition 11) from AAR to $\mathsf{RT}[\cap]$. We assert with the following theorem that the result can be made stronger.

Theorem 5 *There exists a state-matching reduction from the RBAC scheme AAR to* $\mathsf{RT}[\cap]$.

The proof is in the Appendix C.3

4.4 Comparing ATAM with TAM

TAM is a scheme based on the access matrix model and is similar to the HRU scheme [6] (see Section 2.1). Every object is typed, and the type cannot change once the object is created. State-transitions occur via the execution of commands that are similar to HRU commands. We specify a type for every parameter to a command. ATAM is the same as TAM, except that in a condition in an ATAM command, the absence of a right in a cell of the access matrix may be checked (and not just the presence of a right). See Appendices B.5 and B.6 for more details on the two schemes.

Sandhu and Ganta [19] present a mapping from the ATAM to TAM. Based on the mapping, one may conclude that TAM is at least as expressive as ATAM. As the converse is trivially true (TAM is a special case of ATAM), one may conclude that ATAM and TAM have the same expressive power; we gain nothing from the ability to check for the absence of rights in the condition of an ATAM command. Sandhu and Ganta [19] make the observation that the simulation of a command in ATAM may require the execution of an unbounded number of commands in TAM, and conclude with the following comment: "…practically testing for the absence of rights appears to be useful. It is an open question whether this claim can be formalized…" In this section, we formalize this claim by asserting that there is no state-matching reduction from ATAM to TAM.

Theorem 6 *There exists no state-matching reduction from ATAM to TAM.*

The proof is in the Appendix C.4. Thus, the notion of state-matching reductions formalizes the difference in expressive power between ATAM and TAM. One may ask whether there exists a reduction from ATAM to TAM. One may also ask whether reductions or state-matching reductions exist from ATAM to TAM when we allow TAM to contain queries of the type "is $r \notin M_\gamma[s, o]$?" as well (but a command only allows checking for the presence of a right in a cell in the condition). These are open questions.

5 Conclusions and Future Work

We have presented a theory to compare the expressive power of access control models. Our theory is based on perceiving an access control system as a state-transition system, and asking whether there exist security-preserving or strongly security-preserving mappings between two schemes. We have applied our theory in three cases and show that: (1) RBAC with ARBAC97 as its administrative model is limited in its expressive power in comparison to a version of DAC; (2) the trust-management language $\mathsf{RT}[\cap]$ is at least as expressive as RBAC with the URA97 component of ARBAC97 as its administrative model; and (3) ATAM is more expressive than TAM. To our knowledge, (1) is the first evidence that the expressive power of RBAC is limited, and (3) solves an open problem stated in the literature [19].

As future work, we propose to use our theory to compare more models with each other. For instance, we would like to compare various versions of DAC and "layer" these versions based on their relative expressive power. Also, while our theory is based on capturing the notion of policies that can represented and verified in an access control system, we do not believe that reductions and state-matching reductions capture all the types of policies we would want to consider. For instance, a reasonable question to ask during a security audit may be: "did Alice get her write access to a sensitive file only after her husband, Bob was given privileged access to the system?" This can be perceived as a policy issue, and we may want to express this as some expression involving queries. Neither reductions not state-matching reductions capture such query expressions. As part of our future work, we propose to expand our theory to include such policies.

Acknowledgements

Portions of this work were supported by NSF ITR and by sponsors of CERIAS. We thank Ji-Won Byun, Ziad El Bizri, Jiantao Li and Klorida Miraj for their reviews and suggestions on presentation.

References

[1] Paul Ammann, Richard Lipton, and Ravi Sandhu. The expressive power of multi-parent creation in monotonic access control models. *Journal of Computer Security*, 4(2-3):149–166, 1996.

[2] Elisa Bertino, Barbara Catania, Elena Ferrari, and Paolo Perlasca. A logical framework for reasoning about access control models. *ACM Transactions on Information and System Security (TISSEC)*, 6(1):71–127, February 2003.

[3] Ajay Chander, Drew Dean, and John C. Mitchell. A state-transition model of trust management and access control. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 27–43. IEEE Computer Society Press, June 2001.

[4] Srinivas Ganta. *Expressive Power of Access Control Models Based on Propagation of Rights.* PhD thesis, George Mason University, 1996.

[5] G. Scott Graham and Peter J. Denning. Protection — principles and practice. In *Proceedings of the AFIPS Spring Joint Computer Conference*, volume 40, pages 417–429. AFIPS Press, May 16–18 1972.

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

[7] Butler W. Lampson. Protection. In *Proceedings of the 5th Princeton Conference on Information Sciences and Systems*, 1971. Reprinted in ACM Operating Systems Review, 8(1):18-24, Jan 1974.

[8] Ninghui Li and John C. Mitchell. RT: A role-based trust-management framework. In *The Third DARPA Information Survivability Conference and Exposition (DISCEX III)*. IEEE Computer Society Press, April 2003.

[9] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, May 2002.

[10] Ninghui Li and Mahesh V. Tripunitara. Security analysis in Role-Based Access Control, June 2004. To appear in *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT 2004)*.

[11] Ninghui Li, William H. Winsborough, and John C. Mitchell. Beyond proof-of-compliance: Safety and availability analysis in trust management. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 123–139. IEEE Computer Society Press, May 2003.

[12] Ninghui Li, William H. Winsborough, and John C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, February 2003.

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

[14] Sylvia Osborn. Mandatory access control and role-based access control revisited. In *Proceedings of the Second ACM Workshop on Role-Based Access Control (RBAC'97)*, pages 31–40, November 1997.

[15] Sylvia Osborn, Ravi Sandhu, and Qamar Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(2):85–106, May 2000.

[16] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The ARBAC97 model for role-based aministration of roles. *ACM Transactions on Information and Systems Security (TISSEC)*, 2(1):105–135, February 1999.

[17] Ravi Sandhu and Qamar Munawer. How to do discretionary access control using roles. In *Proceedings of the Third ACM Workshop on Role-Based Access Control (RBAC'98)*, pages 47–54, October 1998.

[18] Ravi S. Sandhu. Expressive power of the schematic protection model. *Journal of Computer Security*, 1(1):59–98, 1992.

[19] Ravi S. Sandhu and Srinivas Ganta. On testing for absence of rights in access control models. In *Proceedings of The 6th Computer Security Foundations Workshop*, pages 109–118. IEEE Computer Society Press, June 1993.

A A "Simulation" of RBAC in Strict DAC

We now informally describe a simulation of RBAC in strict DAC, the simplest form of DAC. The point of this simulation is to show that if precise requirements are not specified on simulations, then anything is possible.

The state of a strict DAC model is represented by an access matrix, which has one subject for each user and each role and one object for each permission. There is also one special subject `admin`, who is the creator and owner of every object in the system. All subjects are also objects. We use three rights, '`own`', '`dc`', and '`c`'. We assume that the implementation of the strict DAC model provides the following functionality, it internally sorts all the objects and can return the first object, given an object o, it return the object next to o. The commands implemented in the strict DAC are as follows:

```
command create(s, o)
  create o;
  enter own into (s,o);
end;
command delete(s, o)
  if own ∈ (s,o)
  destroy o;
end;
command grant-dc(s1, s2, o)
  if own ∈ (s1,o)
  enter dc into (s2,o);
  enter c into (s2,o);
end;
command grant-c(s1, s2, o)
  if own ∈ (s1,o)
  enter c into (s2,o);
end;
command revoke-dc(s1, s2, o)
  if own ∈ (s1,o)
  remove c from (s2,o);
end;
command revoke-c(s1, s2, o)
  if own ∈ (s1,o)
  remove c from (s2,o);
end;
```

The addition of new users, roles, and permissions are carried out by the simulator in the straightforward way, i.e., have admin executes a creation command; admin then becomes the owner of these objects. When a new user-role assignment, (u, r), is added, the following procedure is executed, observe that only constant space is needed for the simulation.

```
addUR(u,r) {
 run command grant-dc(admin , u, r);
 while (propagate());
}
propagate() {
 repeat = false;
 for every s,o1,o2 in the matrix {
  if c ∉(s,o2) && c ∈(s,o1) && c ∈(o1,o2) {
   run command grant-c(admin , s, o2);
   repeat = true;
 }}
 return repeat;
}
```

The procedures for adding a role-permission assignment and a role-role inheritance relationship is similar.

Whenever a user-role assignment is removed, the simulator executes the following procedure, which first clear all the propagated rights and redo the propagation.

```
removeUR(u,r) {
 if (dc ∈ (u,r)) {
  run command revoke-dc(admin , u, r);
  clear();
  while (propagate());
 }
}
clear() {
 for every s,o in the matrix {
  if c ∈(s,o) {
   run command revoke-c(admin , s, o2);
 }}
}
```

B Schemes used in Sections 4.2, 4.3 and 4.4

B.1 The SDCO Scheme

Γ SDCO is a scheme based on the access matrix model and is a special case of the HRU scheme (see Section 2.1). Each state $\gamma \in \Gamma$ is $\langle S_\gamma, O_\gamma, M_\gamma[\,], R_\gamma \rangle$ where S_γ, O_γ and R_γ are strict subsets of the countably infinite sets $\mathcal{S}$ (subjects), $\mathcal{O}$ (objects) and $\mathcal{R}$ (rights) respectively. The set of rights for the scheme is $R_\gamma = \{own, r_1, \ldots, r_n\}$, where own is the distinguished right indicating ownership of the object. $M_\gamma[\,]$ is the access matrix.

Ψ The state-transition rules are the commands $createObject$, $destroyObject$ and $grantOwn$, and for each $r_i \in R_\gamma - \{own\}$, a command $grant\_r_i$.

```
command  createObject(s, o)
   create object o
   enter own into [s, o]

command  destroyObject(s, o)
   if own ∈ [s, o]
      destroy o

command  grantOwn(s, s', o)
   if own ∈ [s, o]
      enter own into [s', o]
      remove own from [s, o]

command  grant_r_i(s, s', o)
   if own ∈ [s, o]
      enter r_i into [s', o]
```

Q, $\vdash$ We allow queries of the following forms: (1) Is $s \in S_\gamma$?, (2) Is $o \in O_\gamma$?, and (3) Is $r \in M_\gamma[s, o]$? In these queries, $r \in \mathcal{R}$, $s \in \mathcal{S}$ and $o \in \mathcal{O}$. The entailment relation $\vdash$ is based simply on observing whether the condition holds in the state. For a query of the form (3), if $s \notin S_\gamma$, $o \notin O_\gamma$ or $r \notin R_\gamma$, then γ does not entail that query. This is a natural definition for Q and $\vdash$ for any scheme based on the access matrix model.

B.2 The ARBAC97 Scheme

Γ We assume the existence of the countably infinite sets $\mathcal{U}$ (users), $\mathcal{P}$ (permissions) and $\mathcal{R}$ (roles). An ARBAC97 state is $\langle UA, PA, RH, AR\rangle$ where UA is the user-role assignment relation that contains a pair $\langle u, r\rangle$ for every user $u \in \mathcal{U}$ that is assigned to a role $r \in \mathcal{R}$. PA is the permissions-role assignment relation that contains a pair $\langle p, r\rangle$ for every permission $p \in \mathcal{P}$ that is assigned to the role $r \in \mathcal{R}$. RH is the role-hierarchy, and for $r_1, r_2 \in \mathcal{R}$, $r_1 \succeq r_2 \in RH$ means that all users that are members of r_1 are also members of r_2, and all permissions that are assigned to r_2 are authorized to users that are members of r_1. $AR \subset \mathcal{R}$ is a set of administrative roles. In ARBAC97 [16], changes to AR may be made only by a central System Security Officer (SSO) who is trusted not to leave the system in an undesirable state; if the SSO effects a state-transition, then she does security analysis to ensure that the resulting state is acceptable. Therefore, in our analysis, we assume that AR does not change.

Ψ State-transitions in the ARBAC97 scheme are predicated on the relations that are part of the $URA97$ (user-roles assignment), $PRA97$ (permission-role assignment) and $RRA97$ (role-role assignment) components.

$$URA97 \begin{cases} can\_assign \subseteq AR \times CR \times 2^{\mathcal{R}} \\ can\_revoke \subseteq AR \times 2^{\mathcal{R}} \end{cases} \qquad PRA97 \begin{cases} can\_assignp \subseteq AR \times CR \times 2^{\mathcal{R}} \\ can\_revokep \subseteq AR \times 2^{\mathcal{R}} \end{cases}$$

$$RRA97 \ \{ \ can\_modify \subseteq AR \times 2^{\mathcal{R}}$$

CR is a set of pre-requisite conditions. A pre-requisite condition is a propositional logic formula over regular roles. For instance, $c = r_1 \wedge \overline{r_2}$ is a pre-requisite condition that indicates: "role r_1 and not role r_2," where $r_1, r_2 \in R$.

A state-transition is the successful execution one of the following operations.

```
assignUser(ar, u, rset)
   if ∃ ⟨ar, c, rset⟩ ∈ can_assign such that
   a is a member of ar ∧ u satisfies c ∧
   r ∈ rset then
      assign u to r

revokeUser(ar, u, rset)
   if ∃ ⟨ar, rset⟩ ∈ can_revoke such
   that a is a member of ar ∧
   r ∈ rset then
      revoke u from r
```

```
$assignPermission(a, p, r)$
   $if\ \exists\ \langle ar, c, rset\rangle \in can\_assignp\ such\ that$
   $a\ is\ a\ member\ of\ ar \wedge p\ satisfies\ c \wedge$
   $r \in rset\ then$
      $assign\ p\ to\ r$
```

```
$revokePermission(a, p, r)$
   $if\ \exists\ \langle ar, rset\rangle \in can\_revokep\ such$
   $that\ a\ is\ a\ member\ of\ ar \wedge$
   $r \in rset\ then$
      $revoke\ p\ from\ r$
```

```
$addAsSenior(a, r, s)$
   $if\ \exists\langle ar, rset\rangle \in can\_modify\ such\ that$
   $a\ is\ a\ member\ of\ ar \wedge r, s \in rset\ then$
      $add\ r \succeq s\ to\ RH$
```

```
$removeAsSenior(a, r, s)$
   $if\ \exists\langle ar, rset\rangle \in can\_modify\ such\ that$
   $a\ is\ a\ member\ of\ ar \wedge r, s \in rset\ then$
      $remove\ r \succeq s\ from\ RH$
```

$Q, \vdash$ We allow queries of the following forms that are all natural for the ARBAC97 scheme: (1) given user u, does there exist a role r such that $\langle u, r\rangle \in UA$?, (2) given user u and role r, is $\langle u, r\rangle \in UA$?, (3) given permission p, does there exist a role r such that $\langle p, r\rangle \in PA$?, (4) given permission p and role r, is $\langle p, r\rangle \in PA$?, (5) given roles r_1, r_2, is $r_1 \succeq r_2 \in RH$?, and (6) give user u and permission p, is u authorized to have the permission p? That is, do there exist roles r_1, r_2 such that $\langle u, r_1\rangle \in UA \wedge \langle p, r_2\rangle \in PA \wedge r_1 \succeq r_2 \in RH$? The entailment relation, $\vdash$ is based simply on whether the conditions checked in a query hold in the given state.

B.3 The AAR Scheme

Γ In AAR, a state is the RBAC state $\langle UA, PA, RH\rangle$, as discussed in the previous section.

Ψ The state-transitions allowed are the operations $assignUser$ and $revokeUser$ from the previous section, with the exception that negation is not allowed in pre-requisite conditions. In addition, in AAR, we require that for every role for which there is a can_assign entry, there is also a can_revoke entry. That is, if $\exists\ \langle ar, c, rset\rangle \in can\_assign$ such that ar has at least one member and c may evaluate to $true$, then $\forall\ r \in rset$, $\exists\ \langle ar', rset'\rangle \in can\_revoke$ such that $r \in rset'$ and ar' has at least one member.

$Q, \vdash$ Queries are of the form $s_1 \sqsupseteq s_2$, where s_1 and s_2 are *user-sets*. A user-set is an expression that evaluates to a set of users. A set of roles, a set of permissions and a set of users are user-sets, as are unions and intersections of user-sets. We refer the reader to [10] for more details on user-sets. Entailment involves evaluating the user-sets s_1 and s_2 to the sets of users S_1 and S_2 respectively, and determining whether $S_1 \supseteq S_2$. Several interesting queries related to safety, availability, liveness and mutual-exclusion can be posed as comparisons of user-sets.

B.4 The RT[$\cap$] Scheme

Γ An RT[$\cap$] state is a set of credentials, each of which is one of the following types: (1) $A.r \longleftarrow U$, (2) $A.r \longleftarrow B.r_1$, and (3) $A.r \longleftarrow B.r_1 \cap C.r_2$. Each of A, B, C, U is a principal.

Ψ A state-transition in RT[$\cap$] is either the removal of a credential, or the addition of one. State-transitions are controlled by *growth* and *shrink*-restricted sets of roles — G and S respectively. A role that is in the growth-restricted set may not have any assertions added with that role at the head of the assertion, and a role that is in the shrink-restricted may not have any assertions removed. Thus, the state-transition rules are represented as $\langle G, S\rangle$.

$Q, \vdash$ We allow queries of the form $c_1 \sqsupseteq c_2$ where each c_1 and c_2 is either an RT[$\cap$] role, a credential, or credentials joined by union, $\cup$ or intersection, $\cap$. We observe that this is slightly different from the definition for queries in [10]. The reason is that in that work, only a form-2 weak reduction (see Definition 11) is presented, and therefore queries are processed in conjunction with each state and state-transition rule in the mapping. We seek to map queries independantly of states and state-transition rules. Entailment in RT[$\cap$] is done using credential chain discovery [12]: we find a chain of credentials that proves a (portion of a) query, if one exists.

B.5 The TAM Scheme

Γ TAM is similar to the HRU scheme (see Section 2.1). Each state $\gamma \in \Gamma$ is $\langle S_\gamma, O_\gamma, M_\gamma[\,], R_\gamma, T_\gamma, \mathit{typeOf}\rangle$ where S_γ, O_γ, R_γ and T_γ are strict subsets of the countably infinite sets $\mathcal{S}$ (subjects), $\mathcal{O}$ (objects), $\mathcal{R}$ (rights) and $\mathcal{T}$ (types of objects and subjects) respectively. The function typeOf: $(S_\gamma \cup O_\gamma) \to T_\gamma$, maps each subject and object to a type that cannot change once the subject or object is created. $M_\gamma[\,]$ is the access matrix.

Ψ A state-transition rule is a set of commands. Each command has an optional list of conditions that are joined by conjunction. A command then consists of primitive operations. Each parameter to the command is associated with a type. Each condition may check only for the presence of a right in a cell.

$Q, \vdash$ We allow queries of the form "is $r \in M_\gamma[s, o]$?" Entailment merely involves looking at the contents of the cell to check whether the right is in it. If $r \notin R_\gamma$, $s \notin S_\gamma$ or $o \notin O_\gamma$, then the query is not entailed by γ.

B.6 The ATAM Scheme

$\Gamma, \Psi, Q, \vdash$ An ATAM state is the same as a TAM state. State-transition rules are the same as for TAM, except that a condition in a command may check for the absence of a right (as opposed to only the presence of a right). In ATAM, we allow Q to contain queries of the following two forms: (1) Is $r \in M_\gamma[s, o]$?, and (2) Is $r \notin M_\gamma[s, o]$? This is consistent with the intent of Sandhu and Ganta [19] to determine whether the ability to check for the absence of rights does indeed add more expressive power. $\vdash$ involves merely checking the contents of the relevant cell and determining whether a right is or is not in the cell.

C Proofs for theorems in Section 4

C.1 Proof for Theorem 3

Proof. Sketch: by construction. We have a single administrative role A in ARBAC97 and assign it a single user a. $\mathit{can\_assign} = \{\langle A, \mathit{true}, R\rangle\}$, $\mathit{can\_revoke} = \{\langle A, R\rangle\}$, $\mathit{can\_assignp} = \{\langle A, \mathit{true}, R\rangle\}$, $\mathit{can\_revokep} = \{\langle A, R\rangle\}$, $\mathit{can\_modify} = \{\langle A, R\rangle\}$. There is a user in ARBAC97 corresponding to each subject in SDCO. Each object o in SDCO is associated with the roles $o_{own}, o_{r_1}, \ldots, o_{r_n}$ in ARBAC97. If a user has a right r over the object o, that is mapped to the user being a member of the role o_r. Creation of an object o by subject s in SDCO is mapped to picking the role o_{own} from $\mathcal{R}$ and $\mathit{assignUser}$ of s to o_{own}. Destruction of an object o corresponds to $\mathit{revokeUser}$ of each user from each role corresponding to o. Each query $r \in M_\gamma[s, o]$ is mapped to whether s is a member of the role o_r. Each query $s \in S_\gamma$ is mapped to whether $\langle s, r\rangle \in UA$ for some role r. Each query $o \in O_\gamma$ is mapped to whether $\langle u, o_{own}\rangle \in UA$ for some user u. ∎

C.2 Proof for Theorem 4

Lemma 7 *Let ψ be a state-transition rule, and γ and γ' be states in the $ARBAC97$ scheme. Then, for any two queries q_1 and q_2, there exists no γ' such that $\gamma' \vdash (\neg q_1 \wedge q_2)$ when $\gamma \vdash (q_1 \wedge \neg q_2)$ and $\gamma \mapsto \gamma'$.*

Proof. We observe that the operations $assignUser$, $assignPermission$ and $addAsSenior$ can only cause queries to becomes true, and not false. Similarly, the operations $revokeUser$, $revokePermission$ and $removeAsSenior$ cannot cause a query to become true. Therefore, given a state-transition in the ARBAC97 scheme, it cannot cause a query that is true to become false and another query that is false to become true in the new state. ∎

Proof. (for Theorem 4) By contradiction. Assume that there exists a state-matching reduction from SDCO to ARBAC97. In SDCO, adopt as γ a state with the following properties: $s, s' \in S_\gamma$ with $s \neq s'$, $o \in O_\gamma$ and $own \in M[s, o]$. Let q_1, q_2 and q_3 be queries in SDCO where q_1 is the query "$own \in [s, o]$", q_2 is the query "$own \in [s', o]$" and q_3 is the query "$o \in O_\gamma$". These queries are mapped to q_1^A, q_2^A and q_3^A respectively in the ARBAC97 scheme. We observe that $\gamma \vdash (q_1 \wedge \neg q_2 \wedge q_3)$. There exists a state $\tilde{\gamma}$ reachable from γ such that $\tilde{\gamma} \vdash (\neg q_1 \wedge q_2 \wedge q_3)$. And, there exists no reachable state $\hat{\gamma}$ such that $\hat{\gamma} \vdash (q_1 \wedge q_2 \wedge q_3)$ or $\hat{\gamma} \vdash (\neg q_1 \wedge \neg q_2 \wedge q_3)$ (if $o \in O_\gamma$, then there must be exactly one subject that owns o). Consider the state γ^A in ARBAC97 that corresponds to γ (if there does not exist one, then we have a contradiction). We know that $\gamma^A \vdash \left(q_1^A \wedge \neg q_2^A \wedge q_3^A\right)$. There must also exist a reachable state $\tilde{\gamma}^A$ that corresponds to $\tilde{\gamma}$ (if there does not exist one, then we have a contradiction). By Lemma 7, we know that $\tilde{\gamma}^A$ is not reachable from γ^A is a single state-transition. Therefore, there must exist some state $\hat{\gamma}^A$ that is reachable from γ^A such that $\hat{\gamma}^A \vdash \left(q_1^A \wedge q_2^A \wedge q_3^A\right)$ or $\hat{\gamma}^A \vdash \left(\neg q_1^A \wedge \neg q_2^A \wedge q_3^A\right)$. As there exists no corresponding state in the SDCO scheme that is reachable from γ, we have a contradiction to the assumption that there exists a state-matching reduction from SDCO to ARBAC97. ∎

C.3 Proof for Theorem 5

(This proof appears in [10] as well.)
Proof. By construction. We show that the mapping from [10] from AAR to $\mathsf{RT}[\cap]$ is a state-matching reduction. We consider each assertion from Definition 7 in turn. Each role r in AAR is associated with the role $Sys.r$ in $\mathsf{RT}[\cap]$. We show that after a series of state-transitions, the role-memberships in AAR match the role-memberships in the corresponding state of $\mathsf{RT}[\cap]$.

Assertion 1: Let γ be the given AAR state, and $\gamma \overset{*}{\mapsto}_\psi \gamma'$. Then, $\gamma = \gamma_0 \mapsto_\psi \gamma_1 \ldots \mapsto_\psi \gamma_m = \gamma'$. Each state-transition is either the assignment of a user to a role using $assignUser$ or revocation of a user's membership in a role using $revokeUser$. Let the corresponding states in $\mathsf{RT}[\cap]$ be $\gamma^T = \gamma_0^T, \gamma_1^T, \ldots \gamma_m^T = \gamma^{T'}$. The users that are members of any role r in γ are the same as the users that are members of the corresponding role $Sys.r$ in γ^T. If the state-transition from γ_i to γ_{i+1} is the result of the assignment of the user u to the role r, then we effect the following changes to transition from the state γ_i^T to γ_{i+1}^T: we add the two statements $\mathsf{ASys}.r \longleftarrow u$ and $\mathsf{BSys}.r \longleftarrow u$. If the state-transition is the result of the revocation of the user u from the role r, then we remove all statements that exist of the following two forms: $\mathsf{ASys}.r \longleftarrow u$ and $\mathsf{RSys}.r \longleftarrow u$. We observe that in $\gamma^{T'}$, any $\mathsf{HSys}.r$ has as members all users that were ever members of the role r. Consequently, in $\gamma^{T'}$, each $\mathsf{Sys}.r$ has as members those users that are members of r in γ'. Therefore, we can assert that $\gamma' \vdash q$ iff $\gamma^{T'} \vdash q^T$.

Assertion 2: In $\mathsf{RT}[\cap]$, the only roles that can grow are the ASys and BSys roles. The only roles that can shrink are the ASys and RSys roles. Given $\gamma^T = \sigma(\gamma)$ where γ is a given AAR state and $\gamma^{T'}$ is the corresponding $\mathsf{RT}[\cap]$ state, let $\gamma^T \overset{*}{\mapsto}_\psi \gamma^{T'}$. We construct the AAR state γ' that corresponds to $\gamma^{T'}$ as follows. For each statement of the form $\mathsf{BSys}.r \longleftarrow u$ or of the form $\mathsf{ASys}.r \longleftarrow u$, we assign the user u to the role r. Now, we compare the user-role memberships of each user to the roles r and $\mathsf{Sys}.r$. There cannot be any users in Sys.r that are not in r: the reason is that we have not revoked any user membership in r (starting from the user-role membership in the state γ). There may be users in r that are not in Sys.r. Given the requirement that every role for which there is a can_assign, we also have a can_revoke, the only way for these extra users to be in r and not Sys.r is that there exists a can_assign that permits those users to be assigned to r (starting at the state γ). We revoke such users' membership from r using the relevant can_revoke entries. Now, the memberships in r and Sys.r are identical, and we can assert that for all queries q, $\gamma^{T'} \vdash \sigma(q)$ iff $\gamma' \vdash q$.

∎

C.4 Proof for Theorem 6

Proof. By contradiction. Assume that there exists a state-matching reduction σ from ATAM to TAM. Consider an ATAM scheme in which ψ (the state-transition rule) consists of the following commands.

$$\begin{array}{ll} command\ createSubject(X{:}\ t) & command\ addRight(Y{:}\ t, Z{:}\ t) \\ \quad create\ subject\ X\ of\ type\ t & \quad enter\ r\ into\ [Y, Z] \end{array}$$

Adopt as γ in ATAM a state with the n subjects $s_1, \ldots s_n$, no objects other than subjects, $R_\gamma = \{r\}$, and $T_\gamma = \{t\}$. Let q_i be the query "$r \in M[s_{n+1}, s_i]$" and $\widehat{q_i}$ be the query "$r \notin M[s_{n+1}, s_i]$" for each $i \in \{1, \ldots, n\}$. We first make the following observations:

1. $\gamma \vdash \neg q_1 \wedge \ldots \wedge \neg q_n \wedge \neg \widehat{q_1} \wedge \ldots \wedge \neg \widehat{q_n}$. The reason is that $s_{n+1} \notin S_\gamma$, and therefore, γ does not entail any query related to s_{n+1}. Given this observation, we infer that $\sigma(\gamma) \vdash \neg\sigma(q_1) \wedge \ldots \wedge \neg\sigma(q_n) \wedge \neg\sigma(\widehat{q_1}) \wedge \ldots \wedge \neg\sigma(\widehat{q_n})$. Otherwise, we have the desired contradiction to the existence of a state-matching reduction.

2. there exists $\widetilde{\gamma}$ such that $\gamma \overset{*}{\mapsto}_\psi \widetilde{\gamma}$ and $\widetilde{\gamma} \vdash \neg q_1 \wedge \ldots \wedge \neg q_n \wedge \widehat{q_1} \wedge \ldots \wedge \widehat{q_n}$. $\widetilde{\gamma}$ can be reached from γ by the execution of the command $createSubject$ with the parameter X instantiated with s_{n+1}.

3. for all γ' such that $\gamma \overset{*}{\mapsto}_\psi \gamma'$, $\gamma' \vdash \neg q_i \wedge \neg\widehat{q_i}$ for some $i \in \{1, \ldots, n\}$ if and only if for all $j \in \{1, \ldots, n\}$, $\gamma' \vdash \neg q_j \wedge \neg\widehat{q_j}$. The reason is that any state γ' in which $\neg q_i \wedge \neg\widehat{q_i}$ is true for some $i \in \{1, \ldots, n\}$ is a state in which $s_{n+1} \notin S_{\gamma'}$.

4. there exists no γ' such that $\gamma \overset{*}{\mapsto}_\psi \gamma'$ and $\gamma' \vdash q_i \wedge \widehat{q_i}$ for any $i \in \{1, \ldots, n\}$. The reason is that if $\gamma' \vdash q_i$, then $r \in M_{\gamma'}[s_{n+1}, s_i]$ and therefore $\gamma' \nvdash \widehat{q_i}$.

5. for any $p, q \in \{q_1, \ldots, q_n, \widehat{q_1}, \ldots, \widehat{q_n}\}$, if $p \neq q$ then $\sigma(p) \neq \sigma(q)$. In other words, for σ to be a state-matching reduction, distinct queries in ATAM must map to distinct queries in TAM. Otherwise, let p and q be two distinct queries in ATAM that map to the same query in TAM. Then, in ATAM, there exists γ' such that $\gamma \overset{*}{\mapsto}_\psi \gamma'$ and $\gamma' \vdash p \wedge \neg q$. A corresponding reachable state does not exist in TAM, which gives us the desired contradiction to the existence of a state-matching reduction.

Let $\gamma^A = \sigma(\gamma)$, $\psi^A = \sigma(\psi)$, and $q_i^A = \sigma(q_i)$ and $\widehat{q_i}^A = \sigma(\widehat{q_i})$ for all $i \in \{1, \ldots, n\}$. Given observation (5) above, we know that each query in TAM is distinct from another. Now consider the states that are reachable from γ^A. By observation (2) above, we know that there exists $\widetilde{\gamma}^A$ such that $\gamma^A \overset{*}{\mapsto}_\psi \widetilde{\gamma}^A$ and $\widetilde{\gamma}^A \vdash \neg q_1^A \wedge \ldots \wedge \neg q_n^A \wedge \widehat{q_1}^A \wedge \ldots \wedge \widehat{q_n}^A$. We claim that there also exists a state γ_1^A such that $\gamma^A \overset{*}{\mapsto}_\psi \gamma_1^A$, and γ_1^A has the following property: there exist $i, j \in \{1, \ldots, n\}$ such that $\gamma_1^A \vdash q_i^A \wedge \neg\widehat{q_i}^A \wedge \neg q_j \wedge \neg\widehat{q_j}^A$ or $\gamma_1^A \vdash \neg q_i^A \wedge \widehat{q_i}^A \wedge \neg q_j \wedge \neg\widehat{q_j}^A$. We prove this claim with the following reasoning.

For any $q^A \in \left\{q_1^A, \ldots, q_n^A, \widehat{q_1}^A, \ldots, \widehat{q_n}^A\right\}$, we know from observation (1) above that $\gamma^A \vdash \neg q^A$. There exist states by which queries are entailed that are reachable from γ^A; $\widetilde{\gamma}^A$ is one such state. The only way for q^A to be entailed by a state reachable from γ^A is by a state-transition that adds a right to a cell. This is because the only queries allowed in TAM are those of the form "$r^A \in M^A[s^A, o^A]$". A state-transition (command execution) in TAM can add a right to only a bounded (by a constant) number of cells. Without loss of generality, we assume that this constant is smaller than n. Pick one such state-transition from γ^A that causes either q_i^A or $\widehat{q_i}^A$ to become true, but not q_j^A or $\widehat{q_j}^A$ for some $i, j \in \{1, \ldots, n\}$ with $i \neq j$. We have thus produced the desired state γ_1^A. Note that by observation (4) above, a state-transition cannot cause both q_i^A and $\widehat{q_i}^A$ to become true, because then we have a state in TAM that is reachable from γ^A and entails $q_i^A \wedge \widehat{q_i}^A$, and thus, the desired contradiction to the existence of a state-matching reduction.

By observation (3), a state that entails $q_i \wedge \neg\widehat{q_i} \wedge \neg q_j \wedge \neg\widehat{q_j}$ or $\neg q_i \wedge \widehat{q_i} \wedge \neg q_j \wedge \neg\widehat{q_j}$ does not exist in ATAM. We now have a contradiction to our assumption that there exists a state-matching reduction from ATAM to TAM. ∎