A Theory for Comparing the Expressive Power of Access Control Models[*]

Mahesh V. Tripunitara Ninghui Li
Department of Computer Sciences and CERIAS,
Purdue University, West Lafayette, IN 47905.
{tripunit,ninghui}@cerias.purdue.edu

Abstract

Comparing the expressive power of access control models is recognized as a fundamental problem in computer security. While such comparisons are generally based on simulations between different access control schemes, the definitions for simulations that are used in the literature are informal, and 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 four results. We show that: (1) the well known HRU scheme is limited in its expressive power when compared to a rather simple trust-management scheme, thereby formally establishing a conjecture from the literature; (2) RBAC with a particular administrative scheme from the literature (ARBAC97) is limited in its expressive power, countering claims in the literature that RBAC is more expressive than DAC (Discretionary Access Control) schemes; (3) the ability to check for the absence of rights (in addition to the presence of rights) causes ATAM (Augmented Typed Access Matrix) to be more expressive than TAM (Typed Access Matrix); and (4) a trust-management scheme is at least as expressive as RBAC with a particular administrative scheme (the URA97 component of ARBAC97).

1 Introduction

An access control system enforces a policy on who may access a resource in a certain manner (e.g., "Alice may read the file, f"). The *protection state* (or simply, state) of the system represents all the accesses that are allowed at a given time. 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 the 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 the 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 model* is generally associated with how the state is represented. An example of an access control model is the access matrix model [7, 5, 6], in which a state is represented by a matrix in which each cell, indexed by a $(subject, object)$ pair, contains a set of rights. Formally, an access control model is a set of

[*]A preliminary version of this paper appears in the proceedings of the 2004 ACM Conference on Computer and Communications Security (CCS) [22]

access control schemes; a scheme specifies a set of states, and a set of state-transition rules. An example of a scheme based on the access matrix model is the HRU scheme [6] for which a state is an access matrix, and a state-transition rule is a set of commands, each of which is of a particular form. An *access control system* is an instance of an access control scheme. 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 computer security and is studied extensively in the literature [1, 3, 4, 17, 21, 18, 20]. 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 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 three axes along which these definitions differ.

- The first axis is whether the simulation maps only the state, or also the state-change rule. The approach of Bertino et al. [2] is to map only the states of two access control models to a common language based on mathematical logic, and to compare the results to determine whether one model is at least as expressive as the other, or whether the two models are incomparable. Other work, such as [1, 3, 4, 18, 20] however, require both the state and the state-change rule to be mapped under the simulation.

  An advantage with an approach such as the one that is adopted by Bertino et al. [2] is that it captures "structural" differences in how the protection state is represented in a system based on an access control model. For instance, it is observed in [2] that the existence of an indirection (the notion of a role) between users and permissions in RBAC gives it more expressive power than an access matrix model. Such "structural" differences are not captured by our theory, or other approaches that consider both the state and the state-change rule.

  We point out, however, that the state-change rule is an important component of an access control system, and therefore assert that a meaningful theory for expressive power must consider it as well. In fact, it is often the case that it is the state-change rule that endows considerable power to an access control system. Consider, for example, the access matrix schemes proposed by Graham and Denning [5] and by Harrison et al. [6]. In both schemes, the state is represented by an access matrix. However, the state-change rules are quite different: in the Graham-Denning scheme [5], there are only specific ways in which rights may be transferred, while in the HRU scheme [6], one may define arbitrary commands in a state-change rule. It has also been demonstrated [12] that safety is decidable in polynomial time in the Graham-Denning scheme, while it is known to be undecidable [6] in the HRU scheme. Such differences cannot be captured by an approach that does not consider both the state and the state-change rule.

- The second 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, 20], 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) [17, 21, 15]. 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 third 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. [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 [20]. 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 [17, 15] is qualitatively different from the result that TAM is at least as expressive as ATAM [20], 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. [15, 17, 21] 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 almost all access control models have the same expressive power. Thus, this notion of simulations is not 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, [4] considers this to be a "weak" simulation, while [3] do not consider this to be a simulation at all. Neither decision was justified in [4] and [3].

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.

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 [13, 10], 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 four cases. We show that:

  - there exists no state-matching reduction from a rather simple trust-management scheme, $\mathsf{RT}[\,]$ [10], to the HRU scheme [6]. To our knowledge, this is the first formal evidence of the limited expressive power of the HRU scheme. In [10], Li et al. showed that, contrary to the undecidability result of safety analysis in the HRU scheme, safety analysis and more sophisticated security analysis in the trust management scheme, $\mathsf{RT}[\twoheadleftarrow, \cap]$, is decidable. Li et al. conjectured that these schemes cannot be encoded in the HRU scheme and that the expressive powers of the HRU scheme and of $\mathsf{RT}[\,]$ are incomparable. In this paper, we present formal proof and this.
  - there exists a reduction, but no state-matching reduction from a rather simple DAC scheme, Strict DAC with Change of Ownership (SDCO), to RBAC with ARBAC97 [19] as the administrative model. Several authors [17, 21] have argued that RBAC is more expressive than various forms of DAC, including SDCO. To our knowledge, this is the first evidence of the limited expressive power of an RBAC scheme in comparison to DAC.
  - there exists a state-matching reduction from RBAC with an administrative scheme that is a component of ARBAC97 [19] to $\mathsf{RT}[\cap]$ [8, 9], a trust-management scheme. This shows that state-matching reductions can be constructed for powerful access control schemes in the literature.
  - there exists no state-matching reduction from ATAM to TAM, when we permit queries in ATAM that check for both the absence and the presence of a right in a cell. This revisits the issue addressed by Sandhu and Ganta [20] and formalizes 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, we apply our theory to compare the expressive power of schemes in four cases. We conclude with Section 5. Appendix A presents a “simulation” of RBAC in strict DAC.

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, 20]. 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 [6] in the context of the well-known access matrix model [7, 5]. 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. [6] 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 by [6], simple safety was described as just one class of queries one can consider. Recently, [13, 10] 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, $\vdash: \Gamma \times Q \to \{\mathit{true}, \mathit{false}\}$ is called the entailment relation, and Ψ is a set of state-transition rules.

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

[1]What we call simple safety analysis is called safety analysis in the literature. In [13], 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 [13].

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 reflexive and 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 [11], 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 [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 $\mathcal{S} \subseteq \mathcal{O}$, i.e., all subjects are also objects. 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 $M_\gamma[\,] : S_\gamma \times O_\gamma \rightarrow 2^{R_\gamma}$, 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 $\langle \gamma, \varphi, \psi, \Pi \rangle$, 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. We allow the formula φ to have infinite size. For example, suppose that $\mathcal{S}$, the set of all subjects, is $\{s_1, s_2, s_3, s_4, \ldots\}$, then the formula $\neg(r \in [s_2, o] \vee r \in [s_3, o] \vee r \in [s_4, o] \vee \cdots)$ is true when no subject other than s_1 has the right r over object o.

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 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, $\sigma : (\Gamma^A \times \Psi^A) \cup Q^A \rightarrow (\Gamma^B \times \Psi^B) \cup Q^B$.

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 $\sigma : (\Gamma^A \times \Psi^A) \cup Q^A \rightarrow (\Gamma^B \times \Psi^B) \cup Q^B$, 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, $\langle \gamma^A, \varphi^A, \psi^A, \Pi \rangle$, in A is $\langle \gamma^B, \varphi^B, \psi^B, \Pi \rangle$, 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 $\varphi^B = \sigma(\varphi^A)$. 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 and strongly 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 $\langle\gamma^A, \varphi^A, \psi^A, \Pi\rangle$ in scheme A, let $\langle\gamma^B, \psi^B\rangle = \sigma(\langle\gamma^A, \psi^A\rangle)$ and $\varphi^B = \sigma(\varphi^A)$, we show that $\langle\gamma^A, \varphi^A, \psi^A, \Pi\rangle$ is true if and only if $\langle\gamma^B, \varphi^B, \psi^B, \Pi\rangle$ 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, $\langle\gamma^B, \varphi^B, \psi^B, \exists\rangle$, is also true. On the other hand, if $\langle\gamma^B, \varphi^B, \psi^B, \exists\rangle$ 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 $\langle\gamma^A, \varphi^A, \psi^A, \Pi\rangle$ 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, as the total number of queries may be infinite.) Clearly, φ^A is true in γ_1^A, but $\sigma(\varphi^A)$ 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 $\sigma(\varphi^A)$ 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; it is 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 $\gamma^A \overset{*}{\mapsto}_\psi \gamma_1^A$, 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 $\gamma^B \overset{*}{\mapsto}_{\psi^B} \gamma_1^B$, 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 $\gamma_1^B \vdash^B \sigma(q^A)$.

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 in those models.

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 [18, 20] for simulations.

Definition 10 *(*Form-1 Weak Reduction*)* A mapping from A to B, given by $\sigma : (\Gamma^A \times \Psi^A) \cup Q^A \to (\Gamma^B \times \Psi^B) \cup Q^B$, 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 [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, 20] as 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 $\sigma : \Gamma^A \cup \Psi^A \cup Q^A \rightarrow \Gamma^B \cup \Psi^B \cup Q^B$. 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 $\sigma : \Gamma^A \times \Psi^A \times Q^A \rightarrow \Gamma^B \times \Psi^B \times Q^B$, 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 $\sigma : \Gamma^A \times \Psi^A \times Q^A \rightarrow \Gamma^B \times \Psi^B \times Q^B$ 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 [11] for reductions from security analysis in two RBAC schemes to that in the RT Role-based Trust-management framework [9, 13]. As we state in Theorem 7 in this paper, a form-2 weak reduction used in [11] 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 [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 $\sigma : (\Gamma^A \times \Psi^A) \cup Q^A \rightarrow (\Gamma^B \times \Psi^B) \cup Q^B$, 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 [1], this form of reduction suffices for preserving simple safety properties in 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., [17] 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 [15, 17, 21], 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 the 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, [15] 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 that we now argue to be inadequate 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. Therefore, we argue that it does not make sense to restrict the number of steps involved in the simulation.

Another possibility that we now argue to be inadequate 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, when no restriction is placed on the simulations, the "implementation paradigm" does not separate different access control schemes. On the other hand, it seems difficult to justify the restrictions that have been considered in the literature. Therefore, 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.

4 Applying the Theory

In this section, we apply our theory from Section 2 to compare the expressive power of different access control schemes. In the following section, we show that the HRU access matrix scheme is not as expressive as a relatively simple trust management scheme, $\mathsf{RT}[\,]$. We then examine two particular results from literature using our theory: (1) that RBAC is at least as expressive as DAC (Sections 4.2 and 4.3), and (2) that TAM is at least as expressive as ATAM (Section 4.5), and in each case, assert the opposite. We show also that the trust management scheme $\mathsf{RT}[\cap]$ is at least as expressive as an RBAC scheme (Section 4.4).

Proof Methodology In this section, we prove the existence of reductions and state-matching reductions as well as the nonexistence of state-matching reductions. To prove that there exists a reduction or state-matching reduction from a scheme A to a scheme B, we constructively give a mapping and show that the mapping satisfies the requirements. To prove that there does not exist a state-matching reduction from a scheme A to a scheme B is more difficult, as we have to show that no mapping satisfies the requirements for a state-matching reduction. Our strategy is to use proof by contradiction. We find in scheme A a state γ^A, a state-transition rule ψ^A, as well as a state γ_1^A that is reachable. Suppose, for the sake of contradiction, that a state-matching reduction exists, then there exist states γ^B and γ_1^B such that γ^B is equivalent to γ^A, γ_1^B is equivalent to γ_1^A, and γ_1^B is reachable from γ^B. We show that among the sequence of states leading from γ^B and γ_1^B, there exists one for which there is no matching state that is reachable in A.

4.1 Comparing the HRU scheme to a trust management scheme

The HRU scheme [6] is based on the access matrix model, and has generally been believed to have considerable expressive power, partly because it has been shown that one can simulate a Turing Machine in the HRU scheme. In this section, we show that there does not exist a state-matching reduction from a relatively simple trust management scheme, $\mathsf{RT}[\,]$ [13, 10], to the HRU scheme. That $\mathsf{RT}[\,]$ cannot be encoded in the HRU scheme is informally discussed and conjectured in [13, 10]. Using the theory presented in Section 2, we are able to formally prove this. As safety analysis is efficiently decidable in $\mathsf{RT}[\,]$ but undecidable in the HRU scheme, there does not exist a state-matching reduction from the HRU scheme to the $\mathsf{RT}[\,]$ scheme either. This shows that the expressive powers of the HRU scheme and of $\mathsf{RT}[\,]$ are incomparable. To our knowledge, this Is the first formal evidence of the limited expressive power of the HRU scheme.

The fact that the HRU scheme can simulate Turing Machine shows that it can compute any computable function when used as a computation device. When used as an access control scheme, the HRU scheme may nonetheless be limited in expressive power. For example, it cannot encode an access control system where in one state a subject has no right over any object and in the next state the subject obtains rights over a potentially unbounded number of objects.

The HRU Scheme

Γ We assume the existence of countably infinite sets of subjects, $\mathcal{S}$, objects $\mathcal{O}$ and rights $\mathcal{R}$, with $\mathcal{S} \subset \mathcal{O}$. Each state γ is characterized by $\langle S_\gamma, O_\gamma, R_\gamma, M_\gamma[\,]\rangle$ where $S_\gamma \subset \mathcal{S}$ is a finite set of subjects that exist in the state γ, $O_\gamma \subset \mathcal{O}$ is a finite set of objects that exist in the state γ, $R_\gamma \subset \mathcal{R}$ is a finite set of rights that exist in the state γ, and $M_\gamma[\,]$ is the access matrix, i.e., $M_\gamma[s,o] \subseteq R_\gamma$ gives the set of rights $s \in S_\gamma$ has over $o \in O_\gamma$ in the state γ. $M_\gamma[s,o]$ is defined only when $s \in S_\gamma$ and $o \in O_st$. It may appear that we allow R_γ to differ across states. The definition for state-change rules precludes this possibility.

Ψ A state-change rule, ψ, in the HRU scheme is a command schema, i.e., a set of commands. Each command takes a sequence of parameters, each of which may be instantiated by an object, Each command has also an optional condition, which is a conjunction of clauses. Each clause checks whether a right is in a particular cell of $M_\gamma[\,]$. Following the (optional) conditions in a command is a sequence of primitive operations. The primitive operations are one of the following: (1) create an object; (2) create a subject; (3) enter a right into a cell of the access matrix; (4) remove a right from a cell of the access matrix; (5) destroy a subject; (6) destroy an object. We refer the reader to [6] for more details on the syntax of commands. A state-change is the successful execution of a command.

Q We allow queries of the following two forms: (1) $r \in M[s,o]$, and (2) $r \notin M[s,o]$. In the queries, $r \in \mathcal{R}$, $s \in \mathcal{S}$ and $o \in \mathcal{O}$. To our knowledge, these are the only kinds of queries that have been considered in the context of the HRU scheme in the literature. In particular, these are the queries that are pertinent to the safety property [6].

$\vdash$ Let q be the query $r \in M[s,o]$. Then, given a state γ, $\gamma \vdash q$ if and only if $r \in R_\gamma$, $s \in S_\gamma$, $O \in O_\gamma$ and $r \in M_\gamma[s,o]$. Otherwise, $\gamma \not\vdash q$, or equivalently $\gamma \vdash \neg q$. Let $\widehat{q}$ be the query $r \notin M[s,o]$. Then $\gamma \vdash \widehat{q}$ if and only if $r \in R_\gamma$, $s \in S_\gamma$, $O \in O_\gamma$ and $r \notin M_\gamma[s,o]$. Otherwise, $\gamma \not\vdash \widehat{q}$, or equivalently $\gamma \vdash \neg\widehat{q}$.

Observe that one should view both $r \in M_\gamma[s,o]$ and $r \notin M_\gamma[s,o]$ as atomic queries. In particular $\neg(r \in M_\gamma[s,o])$ is not equivalent to $r \notin M_\gamma[s,o]$. It is possible that $\gamma \not\vdash r \in M_\gamma[s,o]$ and $\gamma \not\vdash r \notin M_\gamma[s,o]$; this happens when either s or o does not exist in γ. Even though it is not possible that $\gamma \vdash ((r \in M_\gamma[s,o]) \wedge (r \notin M_\gamma[s,o]))$.

The RT[ ] Scheme

Γ We assume the existence of countably infinite sets of principals (e.g., A, B, C) and role names (e.g., r, s, t, u). A role is formed by a principal and a role name, separated by a dot (e.g., $A.r, X.u$). An RT[ ] state consists of statements which are assertions made by principals about membership in their roles. Two types of assertions are supported. These are simple member (e.g., $A.r \longleftarrow B$) and simple inclusion (e.g., $A.r \longleftarrow B.r_1$). One reads the $\longleftarrow$ symbol as "includes". The example for the first kind of statement asserts that B is a member of A's r role. The example for the second kind of statement asserts that every member of $B.r_1$ is a member of $A.r$. The portion of a statement that appears to the left of the $\longleftarrow$ symbol is called its head, and the portion that appears to the right is called the body. We refer the reader to [9] for more details on the syntax and semantics of RT[ ] statements.

Ψ A state-change rule in a system based on the RT[ ] scheme consists of two sets, G and S. Both consist of RT[ ] roles. G is the set of growth-restricted roles, i.e., if $A.r \in G$, then statements with $A.r$ at the head cannot be added in future states. S is the set of shrink-restricted roles, i.e., if $A.r \in S$, then roles with $A.r$ at the head cannot be removed in future states. We refer the reader to [13, 10] for more details on the two sets, and the intuition behind them.

Q [13] define three kinds of queries in RT[ ]. (1) $\{B_1, \ldots, B_n\} \sqsupseteq A.r$ – this kind of query asks whether the role $A.r$ is bounded by the set of pricipals $\{B_1, \ldots, B_n\}$; (2) $A.r \sqsupseteq \{B_1, \ldots, B_n\}$ – this kind of query asks whether each principal $B_1, \ldots, B_n$ is a member of $A.r$; (3) $X.u \sqsupseteq A.r$ – this kind of query asks whether the set of member of $A.r$ is included in the set of members of $X.u$.

$\vdash$ Given a state, we check if a query is entailed by first evaluating the set of members of each RT[ ] role in the query. This is done using credential chain discovery [14]. We then compare the two sets and check if the set to the left includes the set to the right. The first two kinds of queries are called semi-static queries as one of the sides in the query is a set of users that is independent of the state, and needs no further evaluation. We refer the reader to [14] for more details on query-entailment in RT[ ].

Theorem 3 *There exists no state-matching reduction from the* RT[ ] *scheme to the HRU scheme.*

Proof. By contradiction. Assume that there exists a state-matching reduction, σ, from the RT[ ] scheme to the HRU scheme. We denote components of a RT[ ] system with the superscript R and the HRU scheme with the superscript H. We now consider a system based on the RT[ ] scheme. Let γ^R be the start-state in our RT[ ] system such that γ^R has no statements. The state-change rule in our RT[ ] system is $G = S = \emptyset$. We now consider the start-state in the corresponding HRU system $\sigma(\gamma^R) = \gamma^H$ and the state-change rule $\sigma(\psi^R) = \psi^H$. Let k be the number of objects in γ^H, i.e., $k = |O_{\gamma^H}|$. Let l be the maximum number of primitive operations of the form "enter right" in any of the commands in ψ^H. Let m be the maximum number of primitive operations of the form "remove right" in any of the commands in ψ^H.

Choose some $n > \big(k^2 + l + m\big) + 1$. Our choice of n is such that for any γ_1^H such that $\gamma^H \mapsto \gamma_1^H$, fewer than $n - 1$ queries that are true in γ^H (i.e., are entailed by γ^H) are false in γ_1^H (i.e., are not entailed by γ_1^H). The reason is that: (1) as γ^H has at most k objects (some or all of which may be subjects), a command may contain statements to destroy all these objects. Consequently, these statements can cause up to k^2 queries of the form $r \notin M[s, o]$ to be false in γ_1^H when they are true in γ^H; (2) as a command in ψ^H has at most l statements to enter rights in to cells, these statements can cause up to l queries of the form $r \notin M[s, o]$ to be false in γ_1^H when they are true in γ^H; (3) as a command in ψ^H has at most m statements to remove rights from cells, these statements can cause up to m queries of the form $r \in M[s, o]$ to be false in γ_1^H when they are true in γ^H. We emphasize that these are the only possibilities for queries to become false in a state-change from γ^H; the number of queries that are entailed by γ^H, but not γ_1^H is fewer than $n - 1$.

Consider queries q_i^R for each integer i such that $1 \le i \le n$ in the RT[ ] system where q_i^R is of the form $\{B_i\} \sqsupseteq A.r$ for some principals $A, B_1, \ldots, B_n$ and some role $A.r$. We make two observations about these queries. The first is that $\gamma^R \vdash q_1^R \wedge \ldots \wedge q_n^R$. The reason is that $A.r$ is empty in γ^R and therefore is a subset of every set of the form $\{B_i\}$. The second observation is that in all states reachable from γ^R, either all queries of the form q_i^R such that $1 \le i \le n$ are entailed, or at most one of those queries is entailed. The reason is that for the set of users in the role $A.r$ to be a subset of $\{B_i\}$ for a particular i, it must be either empty, or contain exactly one element, B_i. Now consider the state γ_t^R such that $\gamma^R \overset{*}{\mapsto}_\psi \gamma_t^R$ and $\gamma_t^R \vdash q_1^R \wedge \neg q_2^R \wedge \ldots \wedge \neg q_n^R$. That is, q_1^R is true in γ_t^R, but none of the other queries of the form q_i^R is true. We use the subscript t only to demarcate the state and not as a count of the number of state-changes needed to reach it. In fact, γ_t^R can be reached from γ^R with a single state-change: we simply add the statement $A.r \longleftarrow B_1$ to our RT[ ] system.

Now consider the corresponding states and queries in the HRU system produced as output by σ. Let $\gamma^H = \sigma(\gamma^R)$, $\gamma_t^H = \sigma(\gamma_t^R)$, and $q_i^H = \sigma(\gamma_i^R)$ for $1 \le i \le n$. As we assume that σ is a state-matching reduction, $\gamma^H \vdash q_1^H \wedge \ldots \wedge q_n^H$, and there exists γ_t^H such that $\gamma^H \overset{*}{\mapsto}_\psi \gamma_t^H$ and $\gamma_t^H \vdash q_1^H \wedge \neg q_2^H \wedge \ldots \wedge \neg q_n^H$. Consider any sequence of state-changes from γ^H to γ_t^H. Pick the first state in the sequence γ_c^H in which at least one of the queries q_i^H is false. Consider the state γ_{c-1}^H immediately preceding it. Then, $\gamma_{c-1}^H \vdash q_1^H \wedge \ldots \wedge q_n^H$. Because one step of change cannot make $n - 1$ queries to go from true to false, in γ_c^H, some queries $q_1, q_2, q_3, \cdots, q_n$ are false but at least 2 queries in them are true. As we argued in the previous paragraph, there cannot exist a matching state in A for γ_c^H. We now have the desired contradiction to the existence of a state-matching reduction from the RT[ ] scheme to the HRU scheme. ∎

4.2 Examining comparisons of RBAC and DAC

[15] present a simulation of ATAM in RBAC and conclude that RBAC is at least as expressive as ATAM. [17, 16, 21] give simulations of various MAC and DAC schemes in RBAC. The main conclusion of [17, 16, 21] 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 [15, 17, 16, 21], the preservation of safety (or other security) properties is not identified as an objective. From the above conclusion in [17, 16, 21], 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 [17, 21] 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.

4.3 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 [15, 17, 21], 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 [19] as the administrative model to that of SDCO, a rather simple form of DAC. We first present precise characterizations of SDCO and the ARBAC97 scheme. We then assert that while there does exist a reduction, 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.

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 finite, 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 $grant\_r_i$.

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

$command\ \ destroyObject(s, o)$
  $if\ \ own \in [s, o]$
    $destroy\ o$

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

$command\ \ grant\_r_i(s, s', o)$
  $if\ \ own \in [s, o]$
    $enter\ r_i\ into\ [s', o]$
```

Q Each query is of one the following forms: (1) Is $s \in S$?; (2) Is $o \in O$?; and (3) Is $r \in M[s, o]$?

$\vdash$ The entailment relation is defined as follows for each type of query from above. In each of the following, $\gamma \in \Gamma$ is a state. (1) $\gamma \vdash s \in S$ if and only if $s \in S_\gamma$; (2) $\gamma \vdash o \in O$ if and only if $o \in O_\gamma$; (3) $\gamma \vdash r \in M[s, o]$ if and only if $r \in R_\gamma \wedge s \in S_\gamma \wedge o \in O_\gamma \wedge r \in M_\gamma[s, o]$.

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 [19], 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 $\mathit{URA97}$ (user-roles assignment), $\mathit{PRA97}$ (permission-role assignment) and $\mathit{RRA97}$ (role-role assignment) components. We introduce the notion of a role range that is used in the definition of the state-transitions. A role range, ξ is written as (r_1, r_2), where r_1 and r_2 are roles, and every role r that satisfies $r_1 \succeq r \wedge r \succeq r_2 \wedge r \neq r_1 \wedge r \neq r_2$ is in the role range ξ. We write $r \in \xi$ when r is in the role range ξ. We represent as Ξ the set of all role ranges. Role ranges in ARBAC97 satisfy some other properties, and we refer the reader to [19] for those. Those properties are not relevant to our discussion here.

$$\mathit{URA97} \begin{cases} \mathit{can\_assign} \subseteq AR \times CR \times \Xi \\ \mathit{can\_revoke} \subseteq AR \times \Xi \end{cases} \qquad \mathit{PRA97} \begin{cases} \mathit{can\_assignp} \subseteq AR \times CR \times \Xi \\ \mathit{can\_revokep} \subseteq AR \times \Xi \end{cases}$$

$$\mathit{RRA97} \; \{ \; \mathit{can\_modify} \subseteq AR \times \Xi$$

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 $r_1, r_2 \in R$.

We postulate that a state-transition is the successful execution one of the following operations.

```
$assignUser(a, u, r)$
  $if\ \exists\ \langle ar, c, \xi \rangle \in can\_assign\ such\ that$
  $a\ is\ a\ member\ of\ ar \wedge u\ satisfies\ c \wedge$
  $r \in \xi\ then$
    $add\ \langle u, r \rangle\ to\ UA$
```

```
$revokeUser(a, u, r)$
  $if\ \exists\ \langle ar, \xi \rangle \in can\_revoke\ such$
  $that\ a\ is\ a\ member\ of\ ar \wedge$
  $r \in \xi\ then$
    $remove\ \langle u, r \rangle\ from\ UA$
```

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

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

```
$addToRange(a, \xi, r)$
  $if\ \exists \langle ar, \xi \rangle \in can\_modify\ such\ that$
  $a\ is\ a\ member\ of\ ar\ then$
    $add\ r_1 \succeq r\ to\ RH$
    $add\ r \succeq r_2\ to\ RH$
    $where\ \xi = (r_1, r_2) \wedge r \neq r_1 \wedge r \neq r_2$
```

```
$removeFromRange(a, \xi, r)$
  $if\ \exists \langle ar, \xi \rangle \in can\_modify\ such\ that$
  $a\ is\ a\ member\ of\ ar\ then$
    $remove\ r_1 \succeq r\ from\ RH$
    $remove\ r \succeq r_2\ from\ RH$
    $where\ \xi = (r_1, r_2) \wedge r \neq r_1 \wedge r \neq r_2$
```

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

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

$Q, \vdash$ We allow queries of the following forms that are all natural for the ARBAC97 scheme: (1) given a role r, does there exist a user u such that $\langle u, r\rangle \in UA$?, (2) given user u, does there exist a role r such that $\langle u, r\rangle \in UA$?, (3) given user u and role r, is $\langle u, r\rangle \in UA$?, (4) given a permission p, does there exist a role r such that $\langle p, r\rangle \in PA$? (5) given permission p, does there exist a role r such that $\langle p, r\rangle \in PA$?, (6) given permission p and role r, is $\langle p, r\rangle \in PA$?, (7) given roles r_1, r_2, is $r_1 \succeq r_2 \in RH$?, and (8) 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.

Theorem 4 *There exists a reduction from SDCO to ARBAC97.*

Proof. By construction. We present the mappings `Reduce` and `ReduceQuery` and show that they satisfy the properties for a reduction from SDCO to ARBAC97. `Reduce` takes as input the start-state and state-transition rules of an SDCO system and produces as output the start-state and state-transition rules of an ARBAC97 system. `ReduceQuery` takes as input a query in the SDCO system and produces as output a query in the ARBAC97 system. We assume, without loss of generality, that there is a one-to-one correspondence between the set of users $\mathcal{U}$ in ARBAC97 and the set of subjects $\mathcal{S}$ in SDCO, and between the set of roles $\mathcal{R}$ in ARBAC97 and the set $(\mathcal{O} \times R_\gamma) \cup \{subjectExists, A, top, bottom\}$, where $\mathcal{O}$ is the set of objects, and R_γ is the set of rights in the SDCO system, and $subjectExists$, A, top and $bottom$ are specific roles that are used in the mapping.

```
Subroutine Reduce($\gamma$,$\psi$)
  /* inputs: $\gamma$ - an SDCO state, $\psi$ - SDCO state-transition rules */
  /* outputs: $\gamma^A$ - an ARBAC97 state,
              $\psi^A$ - ARBAC97 state-transition rules */
  initialize $\gamma^A$,$\psi^A$ as follows:
  $UA = PA = RH = AR = can\_assign = can\_revoke = can\_assignp = can\_revokep = can\_modify = \emptyset$
  add $top \succeq bottom$ to $RH$; let $\xi$ be the role range $(top, bottom)$
  let the set of administrative roles $AR = A$; add $(a, A)$ to $UA$
  $can\_assign = \{\langle A, true, \xi\rangle\}$, $can\_revoke = can\_modify = \{\langle A, \xi\rangle\}$
  execute $addToRange(a, \xi, subjectExists)$ where $subjectExists$ is a role
  foreach $s \in S_\gamma$ execute $assignUser(a, s, subjectExists)$
  execute $removeFromRange(a, \xi, subjectExists)$
  foreach $\langle o, r\rangle \in O_\gamma \times R_\gamma$ execute $addToRange(a, \xi, o_r)$
  foreach $r \in M_\gamma[s, o]$ execute $assignUser(a, s, o_r)$
  return $\gamma^A$,$\psi^A$

Subroutine ReduceQuery($q$)
  /* input: $q$ - an SDCO query */
  /* output: $q^A$ - an ARBAC97 query */
  if $q$ == $s \in S$ then $q^A$ = $\langle s, subjectExists\rangle \in UA$
  if $q$ == $o \in O$ then $q^A$ = $\exists\ u\ such\ that\ \langle u, o_{own}\rangle \in UA$
  if $q$ == $r \in M[s, o]$ then $q^A$ = $\langle s, o_r\rangle \in UA$
  return $q^A$
```

We now show that property (1) for a reduction is satisfied by the above mapping. Let γ_0 be a start-state in SDCO. We produce the corresponding start-state γ_0^A in ARBAC97 using the `Reduce` subroutine above. Given a state γ_k and query q such that $\gamma_0 \overset{*}{\mapsto}_\psi \gamma_k$, we show that there exists γ_k^A and query q^A such that $\gamma_0^A \overset{*}{\mapsto}_{\psi^A} \gamma_k^A$ where $\gamma_k^A \vdash q^A$ if and only if $\gamma_k \vdash q$. If $\gamma_k = \gamma_0$, then $\gamma_k^A = \gamma_0^A$. If q is $s \in S$, then q^A is $\langle s, \mathit{subjectExists}\rangle \in UA$. By line 11 in `Reduce` q^A is true if and only if q is true. If q is $o \in O$, then q^A is $\exists\, u$ such that $\langle u, o_{own}\rangle \in UA$. By line 14 in `Reduce`, and the property that every object that exists in SDCO has an owner associated with it (that is, $own \in M[s,o]$ for some subject s), q^A is true if and only if q is true. And if q is $r \in M[s,o]$, q^A is $\langle s, o_r\rangle \in UA$, and by line 14 of `Reduce`, q is true if and only if q^A is true.

Consider some γ_k reachable from γ_0 and a query q. We show the existence of γ_k^A that is reachable from γ_0^A and that answers q^A the same way by construction. If q is of type $s \in S$, we let $\gamma_k^A = \gamma_0^A$. if q is of type $o \in O$ or $r \in M[s,o]$, we do the following. We consider each state-transition in SDCO $\gamma_0 \mapsto_\psi \gamma_1 \mapsto \ldots \mapsto \gamma_k$. If the state-transition is the execution of $\mathit{createObject}(s,o)$, we execute $\mathit{addToRange}\,(a,\xi,o_{own})$ and $\mathit{assignUser}\,(a,s,o_{own})$. If the state-transition in SDCO is the execution of $\mathit{destroyObject}(s,o)$, we execute $\mathit{revokeUser}\,(a,u,o_r)$ for every $\langle u, o_r\rangle \in UA$ for every r, and $\mathit{removeFromRange}\,(a,\xi,o_{own})$. If the state-transition in SDCO is the execution of $\mathit{grantOwn}\,(s,s',o)$, we execute $\mathit{revokeUser}\,(a,s,o_{own})$ and $\mathit{assignUser}\,(a,s',o_{own})$. If the state-transition in SDCO is the execution of $\mathit{grant\_r_i}\,(s,s',o)$, we execute $\mathit{assignUser}\,(a,s',o_{r_i})$. Now, consider each possible query q. If q is $s \in S$, then $\gamma_k^A = \gamma_0^A$. In our SDCO scheme, the subjects are fixed at the start and never change. So $\gamma_k^A \vdash q^A$ if and only if $\gamma_0 \vdash q$. If q is $o \in O$, then $\gamma_k \vdash q$ if and only if o exists in the state γ_k. This is the case if and only if some subject s has the own right over o. This is the case if and only if we have the role o_{own} in the range ξ and the user corresponding to s is a member of that role. Therefore, $\gamma_k \vdash q$ if and only if $\gamma_k^A \vdash q^A$. And finally, if q is $r \in M[s,o]$, then $\gamma_k \vdash q$ if and only if r has been granted to s by the owner of o. This is true if and only if we have assigned the user corresponding to s to the role o_r. Thus, again, $\gamma_k \vdash q$ if and only if $\gamma_k^A \vdash q^A$.

We prove that property (2) for a reduction is satisfied by our mapping also by construction. Let γ_0^A be the start-state in ARBAC97 corresponding to γ_0, the start-state in SDCO. Then, if γ_k^A is a state reachable from γ_0^A and q^A is a query in ARBAC97 whose corresponding query in SDCO is q, we construct γ_k, a state in SDCO reachable from γ_0 as follows. If q is $s \in S$, we let $\gamma_k = \gamma_0$. Otherwise, for each role o_{own} that has a member s, we execute $\mathit{createObject}\,(s,o)$. For each role o_r that has a member s', if the role o_{own} has a member s, we execute $\mathit{grant\_r}\,(s,s',o)$. If q is $s \in S$, then q^A is $\langle s, \mathit{subjectExists}\rangle \in UA$, and clearly $\gamma_k^A \vdash q^A$ if and only if $\gamma_k \vdash q$, as the subjects that exist do not change from the start-state in SDCO, and the members of $\mathit{subjectExists}$ do not change from the start-state in ARBAC97. If q is $o \in O$, $\gamma_k^A \vdash q^A$ if and only if $\exists\, s$ such that $\langle s, o_{own}\rangle \in UA$. And if q^A is true, we would have added the own right to $M_{\gamma_k}[s,o]$, which means that $\gamma_k \vdash q$ if and only if $\gamma_k^A \vdash q^A$. And finally, if q is $r \in M[s,o]$, $\gamma_k^A \vdash q^A$ if and only if $\langle s, o_r\rangle \in UA$. The condition that q^A is true is the only one in which we would have added the right r to $M_{\gamma_k}[s,o]$, and therefore $\gamma_k \vdash q$ if and only if $\gamma_k^A \vdash q^A$. ∎

Before we introduce Theorem 6, we introduce the following lemma as an intermediate result on the state-change rules in ARBAC97. The intermediate result aids in the proof of the theorem.

Lemma 5 *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 $\mathit{assignUser}$, $\mathit{assignPermission}$, $\mathit{addToRange}$ and $\mathit{addAsSenior}$ can cause queries to become only true, and not false. Similarly, the operations $\mathit{revokeUser}$, $\mathit{revokePermission}$, $\mathit{removeFromRange}$ and $\mathit{removeAsSenior}$ 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. ∎

Theorem 6 *There exists no state-matching reduction from SDCO to ARBAC97.*

Proof. By contradiction. Assume that there exists a state-matching reduction from SDCO to ARBAC97. Let $\mathcal{S} = \{s_1, s_2, s_3, \ldots\}$. In SDCO, adopt as γ a state with the following properties. Let $s_1 \in S_\gamma$, $o \in O_\gamma$ and $own \in M[s_1, o]$. Let q_i be the query "$own \in [s_i, o]$" for each $i = 1, 2, \ldots$, and q_o be the query "$o \in O_\gamma$". These queries are mapped to q_i^A and q_o^A respectively in the ARBAC97 scheme. We observe that $\gamma \vdash (q_1 \wedge \neg q_2 \wedge \neg q_3 \wedge \ldots \wedge q_o)$. There exists a state $\tilde{\gamma}$ reachable from γ such that $\tilde{\gamma} \vdash (\neg q_1 \wedge q_2 \wedge \neg q_3 \wedge \ldots \wedge q_o)$. And, there exists no reachable state $\hat{\gamma}$ such that $\hat{\gamma} \vdash (q_1 \wedge \neg q_2 \wedge \ldots \wedge q_j \wedge \ldots \wedge q_o)$ or $\hat{\gamma} \vdash (\neg q_1 \wedge \neg q_2 \wedge \ldots \wedge \neg q_j \wedge \ldots \wedge q_o)$ for any $j \neq 1$. (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 the desired contradiction). We know that $\gamma^A \vdash \left(q_1^A \wedge \neg q_2^A \wedge \neg q_3^A \wedge \ldots \wedge q_o^A\right)$. There must also exist a reachable state $\tilde{\gamma}^A$ that corresponds to $\tilde{\gamma}$ (if there does not exist one, then we have the desired contradiction). By Lemma 5, 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 $\hat{\gamma}^A \vdash \left(q_1^A \wedge \neg q_2^A \wedge \ldots \wedge q_j \wedge \ldots \wedge q_o^A\right)$ or $\hat{\gamma}^A \vdash \left(\neg q_1^A \wedge \neg q_2^A \wedge \ldots \wedge \neg q_j^A \wedge \ldots \wedge q_o^A\right)$ for at least one $j \neq 1$. 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. ∎

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.

4.4 Comparing an RBAC scheme with a Trust Management Language

In this section, we compare a particular RBAC scheme to the trust management scheme, RT[$\cap$]. The RBAC scheme we consider is called Assignment And Revocation (AAR) [11]. In AAR, the state is an RBAC state, and state-transition rules are those from the URA97 component of the ARBAC97 [19]; users may be assigned to and revoked from roles.

RT[$\cap$] is a trust management scheme 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.

We first present precise characterizations of the AAR scheme and RT[$\cap$]. [11] present a form-2 weak reduction (see Definition 11) from AAR to RT[$\cap$]. We assert with the following theorem that the result can be made stronger.

The AAR Scheme

Γ In AAR, a state is the RBAC state $\langle UA, PA, RH\rangle$, as discussed in the previous section for ARBAC97.

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 $\exists\ \langle ar, c, \xi\rangle \in can\_assign$ such that ar has at least one member and c may evaluate to $true$, then $\forall\ r \in \xi$, $\exists\ \langle ar', \xi'\rangle \in can\_revoke$ such that $r \in \xi'$ 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 [11] 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.

The $\mathsf{RT}[\cap]$ **Scheme**

Γ An $\mathsf{RT}[\cap]$ 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, r, r_1, r_2 is a role name, and $A.r, B.r_1, C.r_2$ is a role. The symbol $\longleftarrow$ is read as "includes". Statement (1) asserts that U is a member of A's r role. Statement (2) asserts that all members of the role $B.r_1$ are members of the role $A.r$. Statement (3) asserts that anyone that is a member of both $B.r_1$ and $C.r_2$ is a member of $A.r$.

Ψ A state-transition in $\mathsf{RT}[\cap]$ 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 $\mathsf{RT}[\cap]$ 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 [11]. 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 independently of states and state-transition rules. Entailment in $\mathsf{RT}[\cap]$ is done using credential chain discovery [14]: we find a chain of credentials that proves a (portion of a) query, if one exists.

Theorem 7 *There exists a state-matching reduction from the AAR scheme to* $\mathsf{RT}[\cap]$.

Proof. By construction. We show that the mapping from [11] 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 $\gamma' \vdash q$. ∎

4.5 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). Below, we present characterizations of the two schemes.

[20] 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. [20] 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.

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 finite, 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 $r \in M[s,o]$?" Entailment is defined as follows. Given a state $\gamma \in \Gamma$, $\gamma \vdash r \in M[s,o]$ if and only if $s \in S_\gamma \wedge o \in O_\gamma \wedge r \in R_\gamma \wedge r \in M_\gamma[s,o]$.

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 $r \in M[s,o]$?, and (2) Is $r \notin M[s,o]$? This is consistent with the intent of [20] to determine whether the ability to check for the absence of rights does indeed add more expressive power. $\vdash$ is defined the same as in TAM for a query of type (1). For a query of the type (2), $\vdash$ is defined as follows. Given a state $\gamma \in \Gamma$, $\gamma \vdash r \notin M[s,o]$ if and only if $s \in S_\gamma \wedge o \in O_\gamma \wedge r \in R_\gamma \wedge r \notin M_\gamma[s,o]$.

Theorem 8 *There exists no state-matching reduction from ATAM to TAM.*

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.

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

Adopt as γ_0 (the start state) in ATAM a state with no subjects or objects. (that is, $S_{\gamma_0} = O_{\gamma_0} = \emptyset$). The set of rights, $R_{\gamma_0} = \{r\}$, and there is a single type t for all subjects (no objects other than subjects exist or can be created in our ATAM system). We denote components of the TAM system under the mapping σ with a superscript T. For example, $\sigma(\gamma_0) = \gamma_0^T$ and $\sigma(\psi) = \psi^T$.

We assume that the countably infinite set of subjects $\mathcal{S} = \{s_1, s_2, \ldots\}$. In the ATAM system, we wish to consider queries of the form $q_{i,j} = r \in M[s_i, s_j]$ and $\widehat{q_{i,j}} = r \notin M[s_i, s_j]$ for some $s_i, s_j \in \mathcal{S}$. First, we make the observation that any two distinct queries $p, q \in \{q_{i,j} | s_i, s_j \in \mathcal{S}\} \cup \{\widehat{q_{i,j}} | s_i, s_j \in \mathcal{S}\}$ are mapped to distinct queries in TAM. That is, $p \neq q \Rightarrow p^T \neq q^T$. Otherwise, pick a pair p, q such that $p \neq q$ but $p^T = q^T$. For any two such queries p and q, there exists a state γ in ATAM such that $\gamma_0 \overset{*}{\mapsto}_\psi \gamma$ and $\gamma \vdash p \wedge \neg q$. Clearly, a corresponding reachable state (that answers the queries p and q the same way) does not exist in TAM, which gives us the desired contradiction. We observe also that by the definition of a state-matching reduction, queries are mapped independent of the start state and the state-change rules.

Consider ψ^T, the command schema in TAM. As a query in TAM is of the form $r \in M[s, o]$, we can determine an upper bound, m, for the number of queries a command in the TAM system can change from false to true when executed. These are queries of both types $q_{i,j}^T$ and $\widehat{q_{i,j}}^T$. One way to determine a value for m is to count the number of "*enter right*" primitive operations in each command and take the maximum (even though this maximum may not be a tight upper bound). m is constant, and may be dependant on γ and ψ, but not the set of queries. Choose some $n > m$.

Now, consider the state in ATAM γ_k such that $\gamma_0 \overset{*}{\mapsto}_\psi \gamma_k$ and $\gamma_k \vdash \neg q_{1,1} \wedge \widehat{q_{1,1}} \wedge \neg q_{1,2} \wedge \widehat{q_{1,2}} \wedge \ldots \wedge \neg q_{n,n} \wedge \widehat{q_{n,n}}$ (we use the subscript k only to distinguish the state, and not as a count of the number of state-changes needed to reach it). That is, γ_k does not entail any of the queries of the type $q_{i,j}$ and entails all queries of the type $\widehat{q_{i,j}}$ for all integers i, j such that $1 \leq i, j \leq n$. The state γ_k corresponds to $S_{\gamma_k} = \{s_1, \ldots, s_n\}$ with no right r in any of the cells. One way to reach this state from γ_0 is to execute the command *createSubject* n times with the parameter instantiated to s_i in the i^{th} execution.

We assume that as σ, a state-matching reduction exists, there exists a corresponding rechable state γ_k^T in TAM that answers the (mapped) queries the same way. Consider any sequence $\gamma_0^T \mapsto_{\psi^T} \gamma_1^T \mapsto_{\psi^T} \ldots \mapsto_{\psi^T} \gamma_k^T$. Pick the first state, γ_c^T in the sequence that satisfies the following condition: $\gamma_c^T \vdash q_{i,j}^T \vee \widehat{q_{i,j}}^T$ for all integers i, j such that $1 \leq i, j \leq n$. Such a state exists: γ_k^T is such a state, and may be the only state in the sequence that meets the condition. We observe also that γ_0^T does not satisfy the condition, thereby implying that the sequence has at least one state-change.

Consider the state γ_{c-1}^T in the sequence just before γ_c^T. γ_{c-1}^T has the following property: there exist integers v, w with $1 \leq v, w \leq n$, such that $\gamma_{c-1}^T \vdash \neg \left(q_{v,w}^T \vee \widehat{q_{v,w}}^T \right) \Rightarrow \gamma_{c-1}^T \vdash \neg q_{v,w} \wedge \neg \widehat{q_{v,w}}^T$. For every state in the ATAM system that entails the corresponding formula of queries $\neg q_{v,w} \wedge \neg \widehat{q_{v,w}}$, the state also entails at least one of the following two formulae of queries: (1) $Q_1 = \neg q_{v,1} \wedge \neg \widehat{q_{v,1}} \wedge \neg q_{v,2} \wedge \neg \widehat{q_{v,2}} \wedge \ldots \wedge \neg q_{v,n} \wedge \neg \widehat{q_{v,n}} \wedge \neg q_{1,v} \wedge \neg \widehat{q_{1,v}} \wedge \ldots \wedge \neg q_{n,v} \wedge \neg \widehat{q_{n,v}}$, or, (2) $Q_2 = \neg q_{w,1} \wedge \neg \widehat{q_{w,1}} \wedge \neg q_{w,2} \wedge \neg \widehat{q_{w,2}} \wedge \ldots \wedge \neg q_{w,n} \wedge \neg \widehat{q_{w,n}} \wedge \neg q_{1,w} \wedge \neg \widehat{q_{1,w}} \wedge \ldots \wedge \neg q_{n,w} \wedge \neg \widehat{q_{n,w}}$.

The reason is that a state in ATAM that entails $\neg q_{v,w} \wedge \neg \widehat{q_{v,w}}$ is one in which either the subject s_v or s_w, or both do not exist ($v = w$ is allowed, and does not affect our arguments). None of the queries of either type $q_{i,j}$ or $\widehat{q_{i,j}}$ corresponding to a subject that does not exist in a state is entailed by the state. Therefore, in TAM, $\gamma_{c-1}^T \vdash Q_1^T \vee Q_2^T$ (where Q_1^T and Q_2^T are obtained from Q_1 and Q_2 respectively by adding the superscript T to each query in the formula).

Consider the state-change in TAM from γ_{c-1}^T to γ_c^T. It must change (at least) n queries that appear in Q_1^T or Q_2^T from false to true. This is not possible, as each state-change can change at most $m < n$ queries from false to true. We have the desired contradiction to the existence of a state-matching reduction from the ATAM scheme to the TAM scheme. ∎

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 $r \not\in M_\gamma[s, o]$?" 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 four cases and show that: (1) the HRU scheme is limited in its expressive power in comparison with a simple trust management scheme; (2) RBAC with ARBAC97 as its administrative model is limited in its expressive power in comparison to a version of DAC; (3) the trust-management scheme RT[$\cap$] is at least as expressive as RBAC with the URA97 component of ARBAC97 as its administrative model; and (4) ATAM is more expressive than TAM. To our knowledge, (1) is the evidence that the expressive of the HRU scheme is limited, (2) is the first evidence that the expressive power of RBAC is limited, and (4) formally demonstrates the benefit from the ability to check for the absence of a right in addition to the presence of a right.

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.

References

[1] Paul Ammann, Richard Lipton, and Ravi S. Sandhu. The expressive power of multi-parent creation in monotonic access control models. *Journal of Computer Security*, 4(2-3):149–165, January 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*, 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, John C. Mitchell, and William H. Winsborough. Beyond proof-of-compliance: Security analysis in trust management, 2004. Accepted to appear in *Journal of the ACM*.

[11] Ninghui Li and Mahesh V. Tripunitara. Security analysis in role-based access control. In *Proceedings of the Ninth ACM Symposium on Access Control Models and Technologies (SACMAT 2004)*, pages 126–135, June 2004.

[12] Ninghui Li and Mahesh V. Tripunitara. On Safety in Discretionary Access Control. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, May 2005.

[13] 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.

[14] 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.

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

[16] 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.

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

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

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

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

[21] Ravi S. 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 1998)*, pages 47–54, October 1998.

[22] Mahesh V. Tripunitara and Ninghui Li. Comparing the expressive power of access control models. In *Proceedings of 11th ACM Conference on Computer and Communications Security (CCS-11)*, pages 62–71. ACM Press, October 2004.

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);
  }}
 }
```