The Next 700 Access Control Models or a Unifying Meta-Model?

Steve Barker
Dept. Computer Science, King's College London
The Strand, London, WC2A 2LS
steve.barker@kcl.ac.uk

ABSTRACT

We address some fundamental questions, which were raised by Atluri and Ferraiolo at SACMAT'08, on the prospects for and benefits of a meta-model of access control. We demonstrate that a meta-model for access control can be defined and that multiple access control models can be derived as special cases. An anticipated consequence of the contribution that we describe is to encourage researchers to adopt a meta-model view of access control rather than them developing the next 700 particular instances of access control models.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Access Controls.

General Terms

Security.

Keywords

Access Control Models, Access Control Policies

1. INTRODUCTION

In his influential paper, entitled "The Next 700 Programming Languages" [17], Peter Landin emphasized that rather than computer scientists developing n special programming languages for n application areas ($n \in \mathbb{N}$), it is essential for them to identify instead a set of programming language "primitives" from which a specific subset may be selected as the basis for deriving a particular language (for a particular area of application).

Over a number of years, researchers in access control have proposed a variety of models and languages in terms of which authorization policies may be defined. Despite the multitude of proposed access control models that have hitherto been described in the literature, we take the view that existing access control models are essentially based on the same (small number of) primitive notions, and that these primitives are interpreted in a limited (and limiting) manner. The essential primitive notions that we identify as common to access control models (and access control specification languages that are based on these models) are: approaches for categorizing entities, methods for describing their properties and the relationships between them, and a means for specifying a range of modalities (of which permission and authorization are of interest, but not exclusively so).

In the approach that we advocate, the primitives of access control that we identify are given a more general interpretation than is usual in access control. Our liberal interpretation will reveal that multiple access control models can be expressed in terms of the primitive notions that we identify, that the degree of overlap amongst existing access control models is significant, and that many "novel" access control models can potentially be developed by simply combining the primitives of access control in novel ways. A consequence of our work is to conclude that research into the universal aspects of access control models should be given prominence rather than the research community continuing to focus on the development of the next 700 particular instances of access control models.

Our work also directly addresses some profoundly important questions in access control, which were raised at SACMAT'08. The specific questions of interest to which we allude are those raised by Atluri and Ferraiolo [12]. By paraphrasing and combining their words, the Atluri-Ferraiolo questions can be expressed thus:

- Is it possible for a unifying access control meta-model to be developed given the large diversity and types of existent access control policies?
- What practical good would such a meta-model serve?

We suggest that the answer to the first question is "yes". In the remainder of the paper, we will give reasons for answering this question in the affirmative. In answer to the second of the Atluri-Ferraiolo questions, we assert that there are many theoretical and practical reasons for developing a unifying access control meta-model. For example, having a general, unifying meta-model of access control is important for providing a common basis for specifying a range of access control notions, which, in turn, facilitates the sharing of access control policy information (including the sharing of policy information by composition). In the ensuing discussion, we will explain more fully the benefits of developing an access control meta-model of the type that we will describe. We will also address potential sources of scepticism about the possibility and potential benefits of a meta-model of access control.

The remainder of this paper is organized thus. In Section 2, we describe the conceptual primitives on which our approach is based: the categorization of principals, the relationships between these categories, and the modalities of relevance in access control. In Section 3, we describe a logic language for describing the meta-model

SACMAT'09, June 3–5, 2009, Stresa, Italy.

of access control that we propose. In Section 4, we show how arbitrary access control policy requirements can be represented in our proposed meta-model by making small changes to the core concepts that constitute our meta-model. We also show how a range of existing access control models and some novel access control models may be viewed as instances of our meta-model. In Section 5, we briefly discuss the "practical good" that a meta-model might deliver. In Section 6, we discuss related work. In Section 7, conclusions are drawn and further work is suggested.

2. FUNDAMENTAL CONCEPTS

In this section, we briefly describe access control in general terms and in relation to the primitive notions of categories, relationships between categories and between categories and principals, and modalities. We also introduce a simple syntax and semantics for discussing these concepts.

Informally, a category (a term which can, loosely speaking, be interpreted as being synonymous with, for example, a type, a sort, a class, a division, a domain) is any of several fundamental and distinct classes or groups to which entities may be assigned. In the approach that we introduce, we regard categories as a primitive concept and we view classification types used in access control, like classifications by role, user attributes, status, clearances, discrete measures of trust, team membership, location, ..., as particular instances of the more general class of category.

It is important to note that we are not concerned with establishing an *a priori* necessarily complete set of categories for access control, and we also only give a descriptive, language-based account of categories. However, the categories that may be used in our meta-model can be arbitrarily complex (e.g., by combining subcategories) and multiple subcategories can be derived from any number of (super-)categories.

In the alphabet that we use to describe our family of access control models, we take the set of entities, which may be referred to in a specification of access control requirements, as a primitive ontological category. Entities are the subjects of predication and cannot themselves be predicated. Entities are denoted uniquely by constants in a many sorted domain of discourse. The key sets of constants in the universe of discourse that we assume in our formulation are as follows:

- A countable set $\mathcal{C}$ of categories, where c_0, c_1, ... are used to denote arbitrary category identifiers.
- A countable set $\mathcal{P}$ of principals, where p_0, p_1, ... are used to identify principals.
- A countable set $\mathcal{A}$ of named atomic *actions*, where a_0, a_1, ... are used to denote arbitrary action identifiers.
- A countable set $\mathcal{R}$ of *resource identifiers*, where r_0, r_1, ... denote arbitrary resources.
- A countable set $\mathcal{S}$ of *situational identifiers*.
- A countable set $\mathcal{E}$ of *event identifiers*, e_0, e_1, ...

Entities in the set $\mathcal{P}$ will include any elements (typically denoted by their public key) that may access a resource in a computer system to which access must be controlled or which may make assertions about other principals (cf. credentials). We assume that principals that request access to resources are pre-authenticated. The actions that we allow are represented by using arbitrary strings of characters that name arbitrary actions that principals may perform; we do not restrict attention to a small, pre-defined set of operations (as is typical in access control models). The situational identifiers that we admit are used to denote contextual or environmental information e.g., IP addresses, times, system states, external states, etc. The precise set $\mathcal{S}$ of situational identifiers that is admitted will, of course, be application specific. On times, we adopt a one-dimensional, linear, discrete view of time, with a beginning and no end point. That is, the model of time that we choose is a total ordering of time points that is isomorphic to the natural numbers. In this paper, we represent times in *YYYYMMDD* format, an encoding of times as natural numbers. Locations and system state indicators are assumed to be represented by strings of characters e.g., "Europe", "System under attack". Event identifiers uniquely denote happenings at a point in time.

It should be noted that although our focus will be on the categorization of principals, categories of other types of entities from our alphabet can be usefully employed in access control. For instance, actions can be viewed in terms of sub-actions, events can be categorized by day of occurrence, contextual information may be used to categorize machines, system states, ..., and a resource can be defined in terms of the sub-resources that it includes.

In addition to the different types of entities that we admit, we consider properties of and relationships between entities.

A property is expressed by a 1-place predicate of the form $p(\tau)$, where τ is a term. For example, $current\_time(t)$ specifies that t has the property of being the current time according to a system clock. Relations are used to describe how one entity may be related to another. Terms of the type that are included in our alphabet may be used in relations. For example, $p(b, d, t)$ may be used to express that $p(b, d)$ holds at time t. Notice too that particular relations may be admitted for particular representations of access control requirements (e.g., ordering relations), and any number of application-specific relations may be defined in order to satisfy domain-specific requirements.

In our approach, the following two relations are of primary importance:

- $\mathcal{PCA}$ is a relation, $\mathcal{PCA} \subseteq \mathcal{P} \times \mathcal{C}$.
- $\mathcal{ARCA}$ is a ternary relation, $\mathcal{ARCA} \subseteq \mathcal{A} \times \mathcal{R} \times \mathcal{C}$.

The semantics of the elements in $\mathcal{PCA}$ and $\mathcal{ARCA}$ are defined thus:

- $(p, c) \in \mathcal{PCA}$ iff a principal $p \in \mathcal{P}$ is assigned to the category $c \in \mathcal{C}$. Henceforth, $pca(p, c)$ will be used to express that principal p is assigned to the category c.
- $(a, r, c) \in \mathcal{ARCA}$ iff the action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$ can be performed by principals assigned to the category $c \in \mathcal{C}$. Henceforth, $arca(a, r, c)$ will be used to express that the a action can be performed on resource r by a principal assigned to category c.

For access control models, two modalities are of prime importance: permissions and authorizations. A permission is a pair (a, r) that denotes that the action a may be performed on resource r. Hence, $arca(a, r, c)$ denotes that the permission (a, r) is assigned to $c \in \mathcal{C}$. An authorization in access control is a principal-assigned permission $\mathcal{PAR}$. That is:

- $\mathcal{PAR}$ is a ternary relation, $\mathcal{PAR} \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{R}$.
- $(p, a, r) \in \mathcal{PAR}$ iff a principal $p \in \mathcal{P}$ can perform the action $a \in \mathcal{A}$ on the resource $r \in \mathcal{R}$. Henceforth, $par(p, a, r)$ will be used to express that the a action on resource r, the permission (a, r), is assigned to the principal p.

The set $\mathcal{PAR}$ is the set of authorizations that hold according to a specification of an access control policy, Π; the set of $par(p, a, r)$ facts that hold with respect to Π may be expressed, in first-order terms, thus:

$$\forall p \in \mathcal{P}\ \forall a \in \mathcal{A}\ \forall r \in \mathcal{R}\ \exists c \in \mathcal{C}[pca(p, c)\ \wedge\ arca(a, r, c) \Rightarrow par(p, a, r)].$$

Access control models will also typically include a relationship ρ between categories that defines (typically) an inclusion relationship between categories c and c'. Hence, par is more generally defined thus:

$$\forall p \in \mathcal{P}\ \forall a \in \mathcal{A}\ \forall r \in \mathcal{R}\ \exists c \in \mathcal{C}\ \exists c' \in \mathcal{C}[pca(p, c) \wedge \rho(c, c')\ \wedge\ arca(a, r, c') \Rightarrow par(p, a, r)].$$

It should be noted that we have used the word "can" when referring to the actions a principal is allowed to perform with respect to a resource. In access control, "can" is standardly interpreted as being synonymous with a principal's possession of a permission. In later sections of this paper we will consider a more general interpretation of "can" than is normal in access control. This general interpretation is required in order to construct the type of general meta-model of access control that we wish to develop.

As a final point on foundational matters, we note that sessions, delegation, denials of permissions and conflict resolution strategies are not considered in the version of the meta-model that we describe in this paper. However, these notions can be naturally accommodated if needs be.

3. RULE LANGUAGE

In this section, we briefly describe a logic language that: will be used by us later in this paper to describe our meta-model; provides a formal semantics for the meta-model; and enables us to specify examples of access control policy representation in terms of the meta-model. Henceforth, we will refer to specifications of access control requirements in our logic language as *access control programs*. We restrict attention to (locally) *stratified programs* [2]. It should be noted that the rule language that we use is quite general and is based on the syntax of constraint logic programs because, as we will see, constraints are important (albeit not necessary) for representing access control requirements in the framework that we describe. Nevertheless, equally serviceable syntactic forms, with equally well defined logical semantics, may be used to specify access control programs in our framework.

We begin by defining the key notions of a *primitive constraint* and a *constraint*.

DEFINITION 3.1. *A primitive constraint c has the form* $p(t_1, \ldots, t_n)$ *where p is a (predefined) constraint relation of arity n and* $t_1, \ldots, t_n$ *are terms. A constraint C is a conjunction of primitive constraints* $c_1 \wedge \cdots \wedge c_k$ *where* $\wedge$ *is the logical 'and' operation.*

We admit equational constraints over sets of constants that denote elements from the alphabet that we adopt. The only terms of interest are constants and variables, and the only primitive constraints of interest are of the form $t_1 = t_2$ and $t_1 \neq t_2$. In the remainder of the paper, constants in our logic language will be denoted by symbols that appear in the lower case; variables are denoted by symbols that start with an upper case letter. We also admit constraints over non-negative integers. Terms are constructed from variables, integer constants and the arithmetic operators $\times, \div, +, -$ and mod; primitive constraints are restricted to the predefined constraint relations $=, \geq, \leq, >, <$ and $\neq$. These relations and the arithmetic operators that we admit are assumed to be predefined in all access control programs.

A rule in an access control program Π takes the following form, which we will later specialize:

$$H \leftarrow L_1, \ldots, L_n.$$

Each rule, $H \leftarrow L_1, \ldots, L_n$, in which L_i $(1 \leq i \leq n)$ is a positive or a negative literal, corresponds to the first-order statement (clause) $\tilde{\forall}(H \leftarrow L_1 \wedge \cdots \wedge L_n)$ where $\tilde{\forall}$ is universal closure.

The semantics of access control programs that we define are expressed in terms of Clark's completion [10].

DEFINITION 3.2. *The definition of an n-place predicate symbol p in an access control program* Π *is the formula*

$$\forall X_1\ \ldots \forall X_n\ p(X_1, \ldots, X_n) \leftrightarrow B_1 \vee \ldots \vee B_m$$

where $\vee$ *is logical or,* $\leftrightarrow$ *is bi-implication and each* B_i *corresponds to a rule in* Π *of the form*

$$p(t_1, \ldots, t_n) \leftarrow L_1, \ldots, L_k.$$

Here, L_i $(1 \leq i \leq k)$ *are literals that may be defined in a local program or in a remotely located program identified by* υ*. In the latter case, we write* $L_i \leftarrow\!\!\!\!\cdot\ \upsilon$*. In the case where a condition of the form* $L_i \leftarrow\!\!\!\!\cdot\ \upsilon$ *appears in a rule in a program* π*, the access control program of interest is the set of rules in* $\pi\ \cup\ \upsilon$*.*

The B_i *elements take the form*

$$\exists Y_1\ \ldots \exists Y_j\ (X_1 = t_1 \wedge \ldots \wedge X_n = t_n \wedge L_1 \ldots \wedge L_k)$$

where $Y_1, \ldots, Y_j$ *are the variables in the original rule, and* $X_1, \ldots, X_n$ *are variables that do not appear in any rule.*

If there is no rule with head p, then the definition of p is simply

$$\forall X_1\ \ldots \forall X_n\ \neg p(X_1, \ldots, X_n).$$

DEFINITION 3.3. *The (Clark) completion,* $\Pi^\star$*, of an access control program* Π *is the conjunction of the definitions of the user-defined predicates in* Π*.*

The following results establish that access control programs are categorical.

PROPOSITION 3.4. *If* Π *is an access control program such that the evaluation of all access requests terminate then* $\Pi^\star$ *has a unique two-valued model.*[1]

The result that follows next demonstrates the coincidence of the logical and an operational semantics for negation-as-failure safe programs [9].

PROPOSITION 3.5. *Let G be an access request and* Π *be an access control program that is negation-as-failure safe for G. Then,* $\Pi^\star \models G \leftrightarrow C$ *iff there are answers* $C_1, \ldots, C_k$ *to G with respect to* Π *(using negation-as-failure) such that* $C \leftrightarrow C_1 \vee \cdots \vee C_k$*.*

In the next two sections, the rule-based language will be used to describe a variety of particular access control models in a common form.

4. AN ACCESS CONTROL META-MODEL

Our meta-model of access control, $\mathcal{M}$, is based on a single core axiom (which is derived from the first-order expression of par from Section 2), to wit:

$$par(P, A, R) \leftarrow pca(P, C), contains(C, C'), arca(A, R, C').$$

[1] *Termination analysis* may be used to ensure termination for all queries of interest. [14]

As the predicate name suggests, $contains$ is a containment relation that expresses that the category C contains the category C'. In terms of principals, the semantics of $contains(C, C')$ is that the set of principals assigned to $C' \in \mathcal{C}$ is a subset of the set of principals assigned to $C \in \mathcal{C}$.

By choosing different definitions of pca, $contains$ and $arca$, the core par axiom of $\mathcal{M}$ can be specialized in multiple ways to define different instances of access control models. It should also be clear that it is perfectly possible for par not to include a $contains$ condition if that is not required for a domain-specific application and it is equally possible for more than one $contains$ relation to be defined in an instance of $\mathcal{M}$.

In the next three subsections, we consider alternative definitions of pca, $contains$ and $arca$ and the different access control models that can be naturally derived from $\mathcal{M}$ by changing their definitions.

4.1 pca Definitions

One way in which a policy author can define access control models in terms of $\mathcal{M}$ is to use pca definitions to define, specialize or combine categories of interest to meet domain-specific requirements. Definitions of pca are specified by using rules of the form defined next.

DEFINITION 4.1. *Definitions of pca are expressed in the form:*

$$pca(P, C) \leftarrow \mathcal{P}_1, \ldots, \mathcal{P}_n, L_1, \ldots, L_p, C_1, \ldots, C_m.$$

Here, $\mathcal{P}_i$ $(1 \leq i \leq n)$ is a condition (possibly negated) that is expressible (recursively) in terms of pca, L_i $(1 \leq i \leq p)$ is an arbitrary literal, and C_i $(1 \leq i \leq m)$ is a sequence of constraints. Any of $\mathcal{P}_1, \ldots, \mathcal{P}_n, L_1, \ldots, L_p$ can be defined at a remotely accessible source or locally. In the former case, the condition is of the form $\mathcal{P}_i \leftarrow\!\!\!\!\wp\ \upsilon$ or $L_i \leftarrow\!\!\!\!\wp\ \upsilon$ where υ is the source of the definition of the literal that appears in the body of a pca rule. $\mathcal{P}_1, \ldots, \mathcal{P}_n$, $L_1, \ldots, L_p$ and $C_1, \ldots, C_m$ are sets of conditions that are disjoint, in a pca rule, ν, and any of these sets may be empty in ν.

Once a category c has been defined by a policy author α, α's definition of c can be used by any number of other policy authors. In particular, if α asserts that a principal p is assigned to a category c then any policy author that sufficiently trusts α's categorization of p as one of α's c's can refer to that category in its specifications of access control requirements.

EXAMPLE 1. *Consider the following policy requirements:*

> *Principals are assigned to the $preferred$ category if they are categorized as being loyal and their current account balance is greater than 1000 Euro (which causes them to be categorized as members of the $goodbalance$ category).*

To represent these requirements by using our approach, it is sufficient to use the following definitions (assuming that all definitions are local):

$$pca(P, pref) \leftarrow pca(P, loyal), pca(P, goodbalance).$$
$$pca(P, goodbalance) \leftarrow balance(P, X), X \geq 1000.$$

Here, $pref$, $loyal$ and $goodbalance$ are domain-specific elements in the general class of categories. □

The important thing to note from the previous example is that *any* categories can be referred to in the generic, meta-model that we are proposing; our proposal is not restricted to particular types of categories. Thus, principals may be assigned to categories according to whether they have a shared attribute, a shared measure of trust, a common security clearance, as a consequence of an assignment to the same department, division, organization, as a consequence of actions or events, ... or any combination of these forms of category types. As access control models are based on category types, it follows that different access control models can be flexibly constructed by combining different types of categories. Moreover, a range of access control concepts can be understood in category-based terms. For example, notions like provisional authorizations (or pre-access obligations) can be represented in category-based terms: a principal that assumes an obligation may be assigned to a category of principals that are obligated to discharge that obligation at some future chronon. It follows from this that we do not distinguish between, for example, what a principal is, has, could be, etc. A principal may be a member of a $manager$ role, have the attribute of being an $adult$, assume an obligation, ... The notion of category is sufficiently powerful to accommodate these particular interpretations. As such, there is no reason to develop 700 access control models to treat these requirements individually; categories are a unifying concept for a meta-model that is capable of accommodating access control requirements, in general.

It should also be noted that although conditions for principal-category assignment can, of course, be expressed in arbitrary (Turing-complete) rule-based access control languages, our motivation is to make categories the basis for access control rather than rules being used to define categories in an ad hoc manner. Several rule-based access control models have been described in the access control literature (see, for example, [4]), but these are just ad hoc variants of the meta-model that we propose in this paper.

The next example that we give illustrates the representation of trusted third-party assertions in our approach.

EXAMPLE 2. *Consider the following policy requirements:*

> *In a local policy specification, any principal P is assigned to the category $approved\_uni$ if a principal Y is assigned to the category of $trusted\_on\_uni$ and Y asserts that P is assigned to the category $good\_university$.*

To represent these requirements, the following definition of pca is sufficient:

$$pca(P, approved\_uni) \leftarrow pca(Y, trusted\_on\_uni), \\ pca(P, good\_university) \leftarrow\!\!\!\!\wp\ Y.$$

□

Many other aspects of access control can be treated, quite generally, within our approach. For example, various access control algebras can be understood as defining particular relationships that exist between the particular types of categories that hold in domain-specific applications for which access control models (and policies) are typically specified. Rather than viewing algebras as a basis for combining policy rules, an algebra of categories allows for rules to be combined and applied to categories of principals. The next definition and example demonstrate how a disjunctive operator can, for example, be simulated and used for specifying particular types of principal-category relationships by using definitions of pca.

DEFINITION 4.2. *Take $\mathcal{P}_{i_j}$ $(1 \leq i \leq n, 1 \leq j \leq k)$ to be a condition expressed in terms of the pca predicate, take L_{i_j} $(1 \leq i \leq p, 1 \leq j \leq k)$ to be literals that are not expressed in terms of pra, take C_{i_j} $(1 \leq i \leq m, 1 \leq j \leq k)$ to be a constraint on $p \in \mathcal{P}$, $c \in \mathcal{C}$ or a term in $L_{1_1}, \ldots, L_{p_1} \ldots L_{1_k}, \ldots, L_{p_k}$, and let Π denote an access control program then*

$$pca(P, C)_1 \leftarrow \mathcal{P}_{1_1}, \ldots, \mathcal{P}_{n_1}, L_{1_1}, \ldots, L_{p_1}, C_{1_1}, \ldots, C_{m_1}$$
$$\vdots$$
$$pca(P, C)_k \leftarrow \mathcal{P}_{1_k}, \ldots, \mathcal{P}_{n_k}, L_{1_k}, \ldots, L_{p_k}, C_{1_k}, \ldots, C_{m_k}$$

expresses that $\forall P \in \mathcal{P}$, P *is assigned to the category* $C \in \mathcal{C}$ *if for some rule* ν *with the head* $pca(P, C)_l$, $(1 \leq l \leq k)$, *the* $\mathcal{P}_i$ *conditions in* ν *are provable from* Π $(\forall i, i \in \{1, .., n\})$, L_q *is provable from* Π $(\forall q, q \in \{1, .., p\})$, *and* C_j *is satisfiable* $(\forall j, j \in \{1, .., m\})$ *with respect to* Π.

EXAMPLE 3. *Principals that are categorized by having a clean driving license (cdl) according to the Driving Vehicle Licensing Authority database (dvla) or have preferred status (ps) are assigned to the "most-valued" customer (mvc) category:*

$$pca(P, mvc) \leftarrow pca(P, cdl) \nleftarrow dvla.$$
$$pca(P, mvc) \leftarrow pca(P, ps).$$

□

For extra convenient expressive power, categories can be parameterized. For instance, $pca(P, manager(b_1))$ may be used as an alternative to

$$pca(P, manager\_b_1).$$

More generally, variables may be used in paramaterized expressions.

4.2 Containment and par Definitions

In the previous section, we considered the flexible specification of access control requirements in terms of pca definitions. In this section, we combine pca definitions and definitions of containment relations to define flexibly a range of particular existing access control models and we show how any number of novel access control models can be represented as specialized instances of our meta-model, $\mathcal{M}$.

In the field of access control, role-based access control (RBAC) has a special importance currently; it has even been speculated that RBAC in itself provides the basis for a meta-model for access control [12]. However, we reject the latter view on the grounds that a role is just a special case of the more general notion of category and, as we will see, RBAC is also a specialized instance of $\mathcal{M}$.

Standard RBAC models [13, 1] assume a single (limited) form of category: the role. In ANSI Hierarchical RBAC, role hierarchies are the only form of category-category relationships that are admitted. In all of the ANSI RBAC models only limited modalities of permissions and authorizations are considered (under a restricted interpretation of "can" cf. Section 2).

In terms of our access control primitives, the axioms that define hierarchical RBAC can be expressed thus (where '_' denotes an anonymous variable):

$$par(P, A, R) \leftarrow pca(P, C), contains(C, C'), arca(A, R, C').$$

$$contains(C, C) \leftarrow dc(C, \_),$$
$$contains(C, C) \leftarrow dc(\_, C),$$
$$contains(C', C'') \leftarrow dc(C', C''),$$
$$contains(C', C'') \leftarrow dc(C', C'''), contains(C''', C'').$$

In this instance, $contains$ is a definition of a partial order relationship between pairs of categories (here restricted to roles) that are in the transitive-reflexive closure of a "directly contains" relation on role identifiers, $dc(r_i, r_j)$, such that: $\Pi \models dc(r_i, r_j)$ iff the role $r_i \in \mathcal{C}$ $(r_i \neq r_j)$ is senior to the role $r_j \in \mathcal{C}$ in an $RBAC$ role hierarchy defined in the access control program Π and there is no role $r_k \in \mathcal{C}$ such that $[dc(r_i, r_k) \wedge dc(r_k, r_j)]$ holds where $r_k \neq r_i$ and $r_k \neq r_j$.

The definition of $contains$ assumes that the following property holds on categories (restricted here to roles):

$$\forall r_i \in \mathcal{C}\ \exists r_j \in \mathcal{C}\ [(dc(r_i, r_j)\ \vee\ dc(r_j, r_i))\ \wedge\ (r_i \neq r_j)].$$

The axiomatization of Hierarchical RBAC models reveals an attractive simplicity of this form of RBAC model. The simplicity appears to be reasonably sufficient for the types of restricted authorization policies that RBAC admits under the simplifying assumptions that it adopts (e.g., principals can be assigned to well-defined, relatively static job functions in "traditional" forms of organizations, . . .). Nevertheless, there are many types of practical access control policies and requirements that need to be represented, but which cannot be adequately expressed in the ANSI RBAC family. Hence, many extended forms of RBAC have been proposed in the access control literature. One such (apparently) extended form of RBAC is *Status-based Access Control (SBAC)* [3].

At first sight, it may appear that SBAC generalizes RBAC by making an important distinction between ascribed status (of which a role assignment is a particular type) and action status. That is, principals can be assigned to a category as a consequence of them being a particular office-holder, but their actions (as office-holders) are also taken into account to determine their overall status. This overall status is used as the basis for determining authorized forms of actions. Thus, ascription is a basis for categorization and so too is the history of an agent's actions. However, ascription and action are simply particular types of category in SBAC. As such, the SBAC extension of RBAC is, in effect, simply one that combines two category types in order for access control decisions to be made. In the meta-model of access control that we propose, *any* number of categories can be so combined and thus many access control models may be accommodated, including SBAC.

To accommodate action status from SBAC, the axioms of our general model of access control may be simply specialized thus (with the above definition of $contains$ assumed and with T - T_s being the interval of time during which a relationship holds):

$$par(P, A, R) \leftarrow pca(P, C), contains(C, C'), arca(A, R, C').$$

$$pca(P, C)\ \leftarrow current\_time(T), happens(E, T_s),\ agent(E, U), act(E, A),\ T_s < T,\ pca\_init(E, P, A, L, T_s, T),\ not\ ended\_pca(P, L, T_s, T).$$

$$ended\_pca(P, L, T_s, T) \leftarrow happens(E', T'),\ agent(E', U),\ act(E', A'),\ pca\_term(E', P, A', L, T_s, T),\ T_s < T',\ T' \leq T.$$

By changing the definitions of pca and $contains$ (and, as we will see later, by changing the definitions of $arca$) new forms of access control models can be derived as particular instances of our meta-model. As an example of that, suppose that a categorization of principals by spatial position were required to determine the principal's set of authorizations. For that, an access control model may be defined as being based, in part, on a categorization of principals by their current location (say). A $contains$ relation may then be defined in terms of dc where dc is used to define geographical regions that are ordered by direct containment (e.g., $dc(europe, uk)$ perhaps). In this case, pca may be defined by using rules of the standard form,

$$pca(P, C) \leftarrow \mathcal{P}_1, \ldots, \mathcal{P}_i, L_1, \ldots, L_m, C_1, \ldots, C_n,$$

but where C' is a categorization of P by location that is expressed by $pca(P, C')$ and where $pca(P, C)$ is one of $\mathcal{P}_1, \ldots, \mathcal{P}_i$. Again, the key point to note is that multiple forms of access control model can be defined as particular cases of $\mathcal{M}$.

Although SBAC may be viewed as a general form of RBAC, MAC and DAC can be viewed as special cases of RBAC (as has

already been noted in the access control literature, see, for example, [21]). In terms of our proposed framework, a version of the Bell-LaPadula model [6] may be viewed as a restricted form of the Hierarchical RBAC model, in which $contains$ is as previously defined and with par defined thus:

$$par(P, read, R) \leftarrow pca(P, C), contains(C, C'), \\ arca(read, R, C').$$

$$par(P, write, R) \leftarrow pca(P, C), contains(C, C), \\ arca(write, R, C).$$

In this case, the containment relationship is an ordering of categories that are restricted to being defined on a common set of security classifications for resources and security clearances for principals. The par definitions represent the rules "no read up" and "write only at the subject's classification level," which are the core axioms of strict MAC (as the latter term is interpreted in Bell-LaPadula terms). The key point to note is that par may be defined as a specialized form of the axiomatization of $\mathcal{M}$.

At this point, another aspect of the generality of $\mathcal{M}$ needs to be considered. Recall that we allow the set $\mathcal{A}$ to include strings of characters that denote arbitrary (atomic) actions. In access control, in general, it is often assumed that $read$ and $write$ are the only actions of interest (cf. strict MAC); access control models that make this assumption are invariably constrained in terms of their expressiveness.

Although there are a many variations, any number of discretionary access control models may also be understood in category-based terms. This should not be too surprising given that groups are a particular type of category and an individual principal p is itself a category: the category that is defined by the singleton $\{p\}$. Delegation via a "with grant option" can also be represented by defining a $contains$ relation as the transitive closure of a form of the $dc(c, c')$ relation, which may be used to specify that c directly delegates permissions to c'. We also note that the discretionary access control model of Unix can be understood in terms of $group$ and $other$ as categories of principals.

Thus far in our discussion, we have considered a variety of basic forms of access control models and their representation in terms of $\mathcal{M}$. Once these basic models have been constructed, they can be specialized further to satisfy a yet wider range of application-specific requirements. For example, $pca(P, C, T_{start}, T_{stop})$ and $arca(P, R, C, T_{start}, T_{stop})$ definitions may be introduced to express intervals of time (i.e., $[T_{start}, T_{stop}]$) during which principal-category assignments and permission-category assignments hold. To accommodate such generalizations, we simply require that the core axioms of $\mathcal{M}$ be specialized too. Specifically, the following form of par may be used:

$$par(P, A, R) \leftarrow current\_time(T), pca(P, C, T', T''), \\ contains(C, C'), arca(A, R, C', T''', T''''), \\ T' \leq T, T \leq T'', T''' \leq T, T \leq T''''.$$

Notions like relative times and periodic times (cf. [7]) can also be naturally defined in order to represent application-specific requirements that are expressible in terms of par.

It should also be noted that par can be defined recursively. This allows for multiple additional access control models to be constructed as particular instances of $\mathcal{M}$. For example, suppose that the domain-specific requirements were for an access control model that combined categorization by status (from SBAC) and categorization by clearance/classification (from MAC). For that, an SBAC program υ_1 may be combined with an MAC program υ_2 by using the following definition:

$$par(P, A, R) \leftarrow par(P, A, R) \hookleftarrow \upsilon_1, par(P, A, R) \hookleftarrow \upsilon_2.$$

As far as access control models based on trust are concerned, we regard the association of a trust measure with a principal as the assignment of the principal to a category of users that have the same degree of trust according to some authority. Moreover, we accommodate assertions made by trusted third parties in our framework by allowing specifications of properties by $p(\tau) \hookleftarrow \upsilon$ and relations by $p(\tau_1, \ldots, \tau_n) \hookleftarrow \upsilon$ (where τ and $\tau_i, i \in \{1, \ldots, n\}$ are terms). We note that, interpreted in terms of certification-based access control, our use of relations of the form $p(\tau_1, \ldots, \tau_n) \hookleftarrow \upsilon$ is essentially the same as principals using n-tuples of the form $p(\tau_1, \ldots, \tau_n)$ to express assertions about public keys via υ (cf. SPKI certificates [11]).

On the RT family of role-trust access control models specifically, we note that the proposers of the RT family [18] begin to address some of the concerns that motivate our paper (concerns on providing a general framework for specifying access control policies). In RT, the notions of roles and "attributes" of principals are used, and trust and role concepts can be combined to allow for a range of access control models to be represented using a common syntactic basis. Nevertheless, roles and trust are simply particular types of category and multiple forms of categories can be combined in $\mathcal{M}$, as we have shown, to accommodate a wider range of access control models than the models admitted in RT.

In RT, a number of general rules are proposed for specifying credentials. For example, the following RT credential (in which $A.r(\tau_1, \ldots, \tau_n)$ and $B.r_1(\sigma_1, \ldots, \sigma_m)$ are roles expressed using the terms $\tau_1, \ldots, \tau_n, \sigma_1, \ldots, \sigma_m$)

$$A.r(\tau_1, \ldots, \tau_n) \leftarrow B.r_1(\sigma_1, \ldots, \sigma_m)$$

has the following equivalent representation in terms of pca definitions (where $C_1, \ldots, C_m$ are constraints on terms that are variables cf. the definition of pca above):

$$pca(P, A.r(\tau_1, \ldots, \tau_n)) \leftarrow pca(P, B.r_1(\sigma_1, \ldots, \sigma_m)), \\ C_1, \ldots, C_m.$$

Other forms of credential that are expressible in the RT syntax can be equivalently represented in terms of the primitives and axiomatization that we admit in $\mathcal{M}$. What is more, we adopt a more general interpretation of category, than the one used in RT, which includes categories defined in terms of events, ticks of a clock, histories of actions, the current location of a requester for access, system states, ...; in short, any category can be admitted in a policy specification expressed in terms of $\mathcal{M}$.

To appreciate further the expressiveness that our proposal affords, consider the following example, which demonstrates how complex access control requirements can be simply represented in $\mathcal{M}$ and how a range of different access control models can be flexibly accommodated in this meta-model.

EXAMPLE 4. *Suppose that the following access control requirements need to be represented:*

> *Any principal that is a member of the category Senior Executive (s_exec), is permitted to read the salary information (as recorded in υ_2) of any principal that is assigned to the category of manager (details recorded locally) of a branch that is categorized as profitable (as recorded in υ_3). To be categorized as a senior executive, a principal must be categorized as being a manager (mgr) of at least five years standing (according to the source of this information, υ_1).*

In this case, it is necessary to deal with various forms of categorization including categorization of an institution (a branch office) having a particular status (of being $profitable$). To represent these requirements, the following specialized axioms of $\mathcal{M}$ are sufficient to include in the policy specification (together with the axioms that define pca, as a 2-place or 3-place predicate, and $salary$ and where $year(T, Y)$ is used to extract the year Y from a time T in YYYYMMDD form):[2]

$$par(P, A, R) \leftarrow pca(P, C), arca(P, R, C).$$
$$pca(P, s\_exec) \leftarrow pca(P, mgr, T) \hookleftarrow \upsilon_1, current\_time(T'), year(T, Y), year(T', Y'), Y' - Y \geq 5.$$
$$arca(read, salary(X, Y), s\_exec) \leftarrow pca(X, manager(Y)), pca(Y, profitable) \hookleftarrow \upsilon_3, salary(X, Y) \hookleftarrow \upsilon_2.$$

□

Any number of additional, novel forms of access control models may be similarly defined in terms of $\mathcal{M}$. For instance, suppose that an access control model were required with a type of $contains$ relationship on categories such that if members of category c_2 trust the assertions of an immediately "superior" authority category c_1 and members of category c_3 similarly trust c_1 then c_2 and c_3 trust each others' assertions (such relationships are often useful in trust-based models [19]). Henceforth, we refer to this access control model as the *Shared Trust Model (STM)*. The (Euclidean) relationship that is required in STM can simply be captured by defining, in terms of dc, a $contains$ relation of the following form:[3]

$$contains(C', C'') \leftarrow dc(C, C'), dc(C, C'').$$

Next, suppose that an access control requirement is such that a principal will engage with whichever principals it chooses to form a *mutual access partnership (MAP)* (cf. policies required in the context of the "policy aware web" [22]). If $p' \in \mathcal{P}$ and $p'' \in \mathcal{P}$ are in a MAP then p' will allow p'' to access its resources and conversely. To represent the MAP model in $\mathcal{M}$, it is sufficient for a policy author to: declare MAP category pairings, using definitions of dc, and to define a symmetrical containment relationship between two principals in a MAP. For the latter, it is sufficient for a policy author to define a rule of the following form:[4]

$$contains(C, C') \leftarrow contains(C', C).$$

It is important to note that the specific details of the STM and the MAP models are relatively unimportant. It is more important to recognize that these models can be naturally represented as instances of $\mathcal{M}$. Moreover, multiple "novel" forms of existing access control models may be similarly developed from the core axioms and predicates of $\mathcal{M}$ (e.g., RBAC with "downward" inheritance of permissions via isa hierarchies of roles).

Next, we note that access control models are often defined, in part, in terms of the classes of constraints that they admit. In $\mathcal{M}$, constraints are expressed in terms of categories.

In Constrained RBAC [13], the constraint of Separation of "Duties" (SoD) is the only general form of constraint that is admitted (albeit static and dynamic versions of SoD are included). A separation of categories (SoC) constraint, which equivalently represents the static SoD constraint in Constrained RBAC, can be specified, in $\mathcal{M}$, in terms of pca, thus (where $\bot$ read as "is inconsistent" and $c \in \mathcal{C}$ and $c' \in \mathcal{C}$ are constants that denote specific categories):

$$\bot \leftarrow pca(P, c), pca(P, c').$$

However, many additional forms of constraints can be similarly represented in $\mathcal{M}$. For example, the following variant of the previous specification of SoC,

$$\bot \leftarrow pca(P, C), pca(P, C'), me(C, C'),$$

can be used to define arbitrary pairs of categories that are mutually exclusive i.e., satisfy the me predicate. We note that a shortcoming of RBAC as a general access control model is that it admits only one restricted form of mutual exclusivity constraint on the one type of category that it assumes, the role. In contrast, in $\mathcal{M}$, mutual exclusivity constraints may be defined with respect to any number of categories.

Many other forms of constraint (beyond SoD/SoC) may be defined in terms of $\mathcal{M}$. For example, the constraint,

$$\bot \leftarrow pca(P, c), not\ pca(P, c'),$$

can be used to express that a principal P cannot be assigned to a category c unless P is assigned to the category c'. That is, a prerequisite constraint can be naturally defined in $\mathcal{M}$. Cardinality constraints may also be defined. Moreover, once the notion of a category is admitted as the basis of $\mathcal{M}$ then constraints on combined category-based access control models are possible. For example, if it were required to define an access control model that combined status categories from SBAC and categories as clearances/classifications from a MAC model then the following constraint can be formulated in $\mathcal{M}$ to specify that no principal with the status of *debtor* (as recorded in the access control program υ_1) can be cleared to access anything other than the resources that are accessible to principals with unclassified clearance (as recorded in the access control program υ_2):

$$\bot \leftarrow pca(P, debtor) \hookleftarrow \upsilon_1, pca(P, C) \hookleftarrow \upsilon_2, C \neq unclassified.$$

Constraints may be similarly defined in terms of pca and can be included in any number of access control models that are derivable from $\mathcal{M}$. Moreover, it is possible to use the language of $\mathcal{M}$ to permit other forms of constraints to be defined. For example, history-based constraints may be defined in terms of events, which, in access control terms, are happenings at an instance of time that typically involve a principal $p \in \mathcal{P}$ (an actor) performing an action in relation to a resource. Thus, to represent that a resource ρ_1 cannot be read more than once on the same day by the same principal (a constraint that is often useful for satisfying the Principle of Least Privilege) the following constraint may be defined in $\mathcal{M}$:

$$\bot \leftarrow happens(E, T), actor(E, P), action(E, read), resource(E, \rho_1), happens(E', T'), actor(E', P), action(E', read), resource(E', \rho_1), E \neq E', T' - T < 1.$$

By judiciously combining the elements of $\mathcal{M}$, any number of specific access control models may be defined in a uniform manner.

4.3 $arca$ Definitions

Thus far, our discussion has been focused on demonstrating how a wide range of access control models can be expressed in terms of our meta-model $\mathcal{M}$ by changing the definitions of pca and $contains$, and hence par. However, there is, as we have mentioned above, a useful generalization of the notion of permissions that is very

[2]Here, the structured terms can be represented in an equivalent "flattened" form if needs be.
[3]More complex forms of this type of Euclidean relation are, of course, clearly possible.
[4]In this case, $contains$ should be a tabled predicate [23] to avoid a potentially infinite computation.

important to adopt in richer interpretations of access control. By adopting this general interpretation, many additional access control models can be defined in terms of $\mathcal{M}$ for satisfying the requirements of domain-specific applications.

To motivate the discussion on this point, consider the language of the *Flexible Authorization Framework (FAF)* [15]. In the FAF, authorizations are represented by the predicates $cando(p, a, r)$ or $dercando(p, a, r)$. In the case of FAF, as is standard in access control, "can do" is interpreted in terms of permission only; FAF does not take alternative interpretations of "can" into account. However, alternative interpretations of "can do" are not only possible, but often need to be represented in access control models in order to capture domain-specific requirements. These alternative interpretations of "can do" are therefore important to accommodate in a meta-model of access control.

In many scenarios it is, for instance, perfectly possible for a principal p to have a permission (a, r), but for p not to be able to do a on r. Moreover, the view of "can do" as synonymous with authorization is not always satisfactory. For example, a server may not have the capability of bringing about a state in which p can do a on r even though p has the permission to do a on r e.g., because p requests to perform an action that cannot be physically satisfied even though it is permitted. It follows that for a richer form of access control meta-model, a more liberal interpretation of "can do", that can capture such nuances, is desirable.

To address the problems of the limited interpretation of "can do" in standard access control, we propose a more general definition of $arca$, in $\mathcal{M}$, than the one that has traditionally been considered. This generalized form of $arca$ also generalizes the notion of authorization (i.e., as principal assigned permissions) that is standard in access control models. Specifically, we advocate defining $arca$ in terms of a range of modalities beyond the interpretation of "can" as permission. For example, in $\mathcal{M}$, "can" may be interpreted in terms of physical capability or in the sense of requiring a willingness on a party, with a resource to protect, to act in order for a requester to perform an action on a resource. Moreover, obligations, in the sense of provisional authorizations, can be understood under a general interpretation of "can". That is, for a principal $p \in \mathcal{P}$ to "do" action a on resource r at time t the "can" requires a willingness by the party that controls access to r to allow access on the basis of p's promise to perform an act at some time point t' such that $t' > t$.

To accommodate these rich interpretations of "can", $arca$ rules may be used and are defined recursively and thus in the same way as pca rules.

DEFINITION 4.3. *Rules defining $arca$ are expressed in the following general form:*

$$arca(A, R, C) \leftarrow \mathcal{A}_1, \ldots, \mathcal{A}_n, L_1, \ldots, L_p, C_1, \ldots, C_m.$$

Here, $\mathcal{A}_i$ ($1 \leq i \leq n$) is a condition that is expressible (recursively) in terms of $arca$, L_i ($1 \leq i \leq p$) are literals, and C_i ($1 \leq i \leq m$) is a sequence of constraints. Any of $\mathcal{A}_1, \ldots, \mathcal{A}_n, L_1, \ldots, L_p$ can be defined at a remotely accessible source or locally. In the former case, the condition is of the form $\mathcal{A}_i \hookleftarrow \upsilon$ or $L_i \hookleftarrow \upsilon$ where υ is the source of the definition of the literal that appears in the body of an $arca$ rule. $\mathcal{A}_1, \ldots, \mathcal{A}_n$, $L_1, \ldots, L_p$ and $C_1, \ldots, C_m$ are sets of conditions that are disjoint, in a $arca$ rule, ν, and any of these sets may be empty in ν.

It should be clear from the general definition of $arca$ above that different interpretations of this predicate can be flexibly employed in a variety of different ways and, as a consequence, any number of additional access control models can be defined as instances of $\mathcal{M}$. The following example illustrates the possibilities afforded by a generalized interpretation of $arca$.

EXAMPLE 5. *Consider the following policy requirements:*

> *A principal's request to buy gold is permitted (in the sense of being physically possible) provided that the amount of gold requested is not greater than the current stock level recorded in υ_1. In a gold market that is currently categorized as "volatile", according to the source υ_2, a principal that requests to perform an act of buying is permitted to buy a maximum of 50 units of gold (i.e., permission as consistency with supplier intentions). All principals are permitted (in the sense of being authorized) to perform a buying action in relation to the resource gold provided that the principal is not a member of the debtor category.*

To represent these access control requirements in our framework, the following rules may be used:

$$arca(A, R, C) \leftarrow arca_c(A, R, C), arca_p(A, R, C), \\ not\ arca_i(A, R, C).$$
$$arca_c(buy, gold(X), C) \leftarrow stock(gold, Y) \hookleftarrow \upsilon_1, \\ Y - X \geq 0.$$
$$arca_p(buy, gold(X), C) \leftarrow C \neq debtor.$$
$$arca_i(buy, gold(X), C) \leftarrow X > 50, \\ market(gold, volatile) \hookleftarrow \upsilon_2.$$

In this case, if a request is received from a principal p to perform the action of buying from a principal p' that defines the access control program above then that action is allowed if and only if p' has the capability of satisfying p's request, p' permits the request, in the sense of authorizing p to perform the action of buying gold, and p' has the intention of satisfying p's request given the particular state of the gold market that obtains. □

It is important to note, from the previous example, that the access control program is based on one possible distinction between the different interpretations of "can." Other interpretations are, of course, possible and can be used to define different instances of $\mathcal{M}$. It is also important to note that the different interpretations of "can", which are used in the example above, cannot simply be captured by merging conditions into a single rule in a rule-based approach to access control requirement representation. For instance, physical capability does not demand that a particular state of the market obtains. The separate $arca$ definitions here emphasize that different aspects of "can" need to be separately specified.

In $\mathcal{M}$, any number of constraints may be expressed in terms of $arca$. For example,

$$\perp \leftarrow arca(write, o_1, c), not\ arca(write, o_1, c'),$$

may be used to specify a "prerequisite" constraint on permissions, to wit: for the $write$ action to be performed on the resource o_1 by principals assigned to the category c it is required that the $write$ action on o_1 is assigned to principals assigned to the category c'.

Notice too that if multiple interpretations of "can" are accommodated in a meta-model like $\mathcal{M}$ then it is possible to specify general forms of constraints in terms of the variants of $arca$ that are admitted. For example,

$$\perp \leftarrow arca_c(A, R, C), not\ arca_p(A, R, C),$$

may be used to represent the constraint that, for all categories of principals, it is impossible for a permission not to be assigned to a category if any requested action A on any resource R is (physically) capable of being performed.

As a final point of $arca$ definitions, we note that a variety of different interpretations of denials of access to categories of principals can be naturally accommodated in an extended form of $\mathcal{M}$. Due to

space limitations, we do not give the details of that here. However, it should be clear that a d_arca predicate (say) could be used to define denials of permission assignments to categories of principals in essentially the same way that we have used $arca$ definitions (and with different interpretations of denials being admitted e.g., denials by intention, denials as physical constraints, etc.).

5. THE PRACTICAL GOOD

Thus far, we have focused on one of the Atluri-Ferraiolo questions, the question: is it possible to define an access control meta-model? In this section, we briefly address the second of the Atluri-Ferraiolo questions: what "practical good" would a meta-model serve?

Following on from the discussion in Section 1 of this paper, having a general, unifying access control meta-model provides a basis for a common specification of access control requirements that permits a general semantics to be determined for access control. Having a common, agreed semantics is essential when access control information needs to be shared (as is the case with many distributed applications). The meta-model $\mathcal{M}$ that we have described has a well-defined semantics (which can be expressed in terms of Clark's completion or perfect models). By ensuring that all policies that are defined in terms of $\mathcal{M}$ are (locally) stratified, properties of these policies can be directly determined (e.g., from standard results for the operational semantics used for access request evaluation). Moreover, having a meta-model that is based on well defined formal foundations and common semantics facilitates the sharing of access control information.

Having a shared conception of an access control meta-model is important for reducing the burden on policy administrators when it comes to representing application-specific access control requirements. That is, the meta-model provides policy authors with a general framework for representing access control requirements. A policy author simply needs to specialize the general axioms to define a domain-specific model and the access control policies that can be represented within that model. Because the meta-model reduces the burdens on policy authors (i.e., it provides a well defined framework for policy authors to specialize), it is useful for the rapid prototyping of policies and for abstracting away the complexities of access control policy specification.

Identifying a common access control model is also desirable because it allows for various general syntaxes to be developed in terms of the generic model e.g., a natural language syntax for simplifying access control specification and an XML-based syntax for access control policy exchange. In relation to an XML-based syntax for access control policy exchange, we argue that access control policies that are expressible in our meta-model can be naturally represented in RuleML [8]. A representation of our meta-model in RuleML offers the possibility of access control policy exchange in a framework that has a well defined formal semantics (unlike XACML).

6. RELATED WORK

In response to our proposed approach and focus, a sceptical reader may argue that general models and languages have already been described in the access control literature e.g., the Generalized TRBAC model [16] and ASL [15]. However, it is our view that these approaches, though valuable in their own right, cannot be meaningfully described as general in any absolute sense; they are general only in the sense of being more general than the particular access control models that they assume as a primitive base. In GTRBAC, the focus is on one type of category, the role (interpreted as being synonymous with the notion of job function). In ASL, users, groups and roles (again, traditionally interpreted in functional terms) are admitted in the language. FAF/ASL could, of course, be extended to some extent to accommodate the richer notion of categories that we advocate using. However, a richer form of ASL/FAF would be required to accommodate the generality of $\mathcal{M}$ e.g., to allow for hierarchies that are not restricted to partial orders, for assertions made by remote authorities, for an extended form of $done$ that, for instance, admits proactive events and more expressive forms of event descriptions, for a generalized interpretation of "can", etc.

From the discussion above, it will be clear that, despite the extensive literature on RBAC, it is our view that RBAC is simply a particular instance of $\mathcal{M}$. More strongly, RBAC is not even an especially significant instance of $\mathcal{M}$ for it is based on a single, semantically impoverished category (the role), one $contains$ relation (a partial ordering of roles), and one type of constraint, a SoD constraint. The proponents of RBAC point out that the elements of RBAC can be flexibly combined to allow for a range of RBAC models and policies to be defined, but the concepts included in RBAC are not always sufficiently expressive to enable domain-specific requirements to be captured even by combination. In the case where RBAC is not sufficiently expressive to represent requirements, ad hoc extensions may be employed but these extensions are simply particular instances of $\mathcal{M}$ and may compromise the sharability of access control policy information. Although it remains a useful special case of an access control model that is applicable in certain contexts, RBAC is not a sufficiently general model of access control; rather, RBAC is a special case of $\mathcal{M}$.

A sceptical reader might also argue that a general *language* for access control policy specification has already been described in the access control literature: XACML [20]. However, in our view, it is essential to define a general access control language in terms of a well-defined access control model with a sound formal semantics (rather than developing ad hoc access control languages with hopelessly inadequate formal semantics, as is the case with XACML). In addition to its unsatisfactory formal underpinnings, XACML is not based on a well defined conceptual model of access control. Attempts to retrofit aspects of access control models (via profiles) have not been satisfactory.

The sceptical reader might also argue that there has never been a universal agreement on a "general" programming language and there is therefore no reason to think that a general access control model/language needs to be considered. However, although many programming languages have been developed for many different applications, these languages do have common features that derive from a general model of computation that they assume (cf. Landin's work [17]).

The sceptic may also argue that any access control model that is claimed to be general, as RBAC has been suggested to be [4], will necessarily end up having numerous ad hoc features, in order to make it generally applicable, and will thus be necessarily complex as a consequence. However, we have shown that, by applying Ockham's razor to the previously developed 700 access control models, a small core set of primitives can be identified that, despite the limited concepts involved, paradoxically provides considerable expressive power that obviates the need for multiple ad hoc features.

The work by Li et al. [18] is related to ours in several respects. Li et al.'s RT family of role-trust models provides a quite general framework for defining access control policies, it includes some standard syntactic forms that can be specialized for defining specific policy requirements (in terms of credentials), and it is based on a well defined formal semantics, which permits properties of policies to be proven. Our approach, however, includes concepts

like times, events, actions and histories that may be used to specify principal-category assignments, but which are not included as elements of RT. Moreover, in RT the focus is on a particular types of categorization of principals: by their role membership or their attributes. In $\mathcal{M}$, we allow for a richer range of categories (e.g., we allow for obligations to be treated using categories, for principal categorization according to histories of actions, ...), we consider categories in relation to permission-category assignments as well as principal-category assignments, and our permission-category assignments are based on a more general interpretation of "can do" than that that is standard in access control. In RT, some particular forms of rules are included as part of the model. In contrast, we allow rules for defining pca, $arca$ and par in (locally) stratified logic in general. As such, in $\mathcal{M}$, it is possible to specify access control requirements in terms of the non-assignment of principals to a category, for example.[5]

The work on SecPAL [5] is motivated, as ours is, by the desire to define a general, declarative framework for specifying a wide range of authorization policies. However, in SecPAL the emphasis is on a language for realising this goal. In our approach, a general underlying model is the focus of study and the language requirements are derived directly from the meta-model.

7. CONCLUSIONS AND FURTHER WORK

In this paper, we have addressed two profoundly important questions in access control, the Atluri-Ferraiolo questions:

- Is it possible for a unifying access control meta-model to be developed given the large diversity and types of existent access control policies?
- What practical good would such a meta-model serve?

Taking these questions in reverse order, we have described several theoretical and practical benefits (e.g., on policy sharing, see Section 5) that can be realised if the meta-model of access control, $\mathcal{M}$, were to be adopted. In response to the first question, we have shown that it is possible to define a meta-model of access control that is based on the primitive notion of categories, relationships between categories and principals and amongst categories, and a notion of "can do", and hence authorization, that is more general than the one that is usually adopted in access control. We have also discussed how a wide range of constraints may be expressed in $\mathcal{M}$ and how category-based algebras may be defined and used for access policy composition. For specifications of access control policies in terms of $\mathcal{M}$, only four predicates and a standard axiomatization of "authorization", that can be flexibly specialized, are required. From this simple base, it is possible to reformulate the last 700 access control models that have been developed and to develop 700 additional novel forms of access control model.

Future work includes to generalize $\mathcal{M}$ to, for instance, accommodate a richer variety of different interpretations of denials than those that have previously been considered by researchers in access control (and to then consider appropriate conflict resolution strategies). We also intend to investigate the development of a natural language and a markup language for the exchange of access control policies expressed in terms of $\mathcal{M}$. Moreover it is important to note that the meta-model that we have proposed still makes it possible to define many potentially interesting access control models as special cases of the meta-model. The investigation of additional, specialized forms of $\mathcal{M}$ is also a matter for further work.

[5]However, although we admit an nonmonotonic negation operator, we assume that all sources of access control information are complete.

8. REFERENCES

[1] ANSI. RBAC, 2004. INCITS 359-2004.

[2] C. Baral and M. Gelfond. Logic programming and knowledge representation. *JLP*, 19/20:73–148, 1994.

[3] S. Barker, M. J. Sergot, and D. Wijesekera. Status-based access control. *ACM Trans. Inf. Syst. Secur.*, 12(1), 2008.

[4] S. Barker and P. Stuckey. Flexible access control policy specification with constraint logic programming. *ACM Trans. on Information and System Security*, 6(4):501–546, 2003.

[5] M. Y. Becker, C. Fournet, and A. D. Gordon. Design and semantics of a decentralized authorization language. In *CSF*, pages 3–15, 2007.

[6] D. E. Bell and L. J. LaPadula. Secure computer system: Unified exposition and multics interpretation. *MITRE-2997*, 1976.

[7] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. *ACM TODS*, 23(3):231–285, 1998.

[8] H. Boley, S. Tabet, and G. Wagner. Design rationale of ruleml: A markup language for semantic web rules. In *SWWS 2001*, pages 381–401, 2001.

[9] L. Cavedon and J. Lloyd. A completeness theorem for SLDNF resolution. *JLP*, 7(3):177–192, 1989.

[10] K. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322. Plenum, 1978.

[11] D. E. Clarke, J.-E. Elien, C. M. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in SPKI/SDSI. *J. Computer Security*, 9(4):285–322, 2001.

[12] D. F. Ferraiolo and V. Atluri. A meta model for access control: why is it needed and is it even possible to achieve? In *SACMAT'08*, pages 153–154, 2008.

[13] D. F. Ferraiolo, R. S. Sandhu, S. I. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed nist standard for role-based access control. *ACM TISSEC*, 4(3):224–274, 2001.

[14] S. Genaim and M. Codish. Inferring termination conditions for logic programs using backwards analysis. volume 2250 of *LNCS*, pages 685–694. Springer, 2001.

[15] S. Jajodia, P. Samarati, M. Sapino, and V. Subrahmaninan. Flexible support for multiple access control policies. *ACM TODS*, 26(2):214–260, 2001.

[16] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, 17(1):4–23, 2005.

[17] P. J. Landin. The next 700 programming languages. *Commun. ACM*, 9(3):157–166, 1966.

[18] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *IEEE Symposium on Security and Privacy*, pages 114–130, 2002.

[19] C.-J. Liau. Belief, information acquisition, and trust in multi-agent systems–a modal logic formulation. *Artif. Intell.*, 149(1):31–60, 2003.

[20] OASIS. eXtensible Access Control Markup language (XACML), 2003. http://www.oasis-open.org/xacml/docs/.

[21] R. S. Sandhu and Q. Munawer. How to do discretionary access control using roles. In *ACM Workshop on Role-Based Access Control*, pages 47–54, 1998.

[22] D. J. Weitzner, J. Hendler, T. Berners-Lee, and D. Connolly. Creating a policy-aware web: Discretionary, rule-based access for the world wide web. *Web and Information Security*, 2006.

[23] *The XSB System Version 2.7.1, Programmer's Manual*, 2005.