Steve Barker
Guido Boella
Dov M. Gabbay
Valerio Genovese

A Meta-model of Access Control in a Fibred Security Language

Abstract. The issue of representing access control requirements continues to demand significant attention. The focus of researchers has traditionally been on developing particular access control models and policy specification languages for particular applications. However, this approach has resulted in an unnecessary surfeit of models and languages. In contrast, we describe a general access control model and a logic-based specification language from which both existing and novel access control models may be derived as particular cases and from which several approaches can be developed for domain-specific applications. We will argue that our general framework has a number of specific attractions and an implication of our work is to encourage a methodological shift from a study of the particulars of access control to its generalities.

Keywords: Logic, Fibring, Access Control, Security.

Introduction and overview

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 (see, for example, [6, 8, 21, 11]).

In this paper, we argue that existing access control models are based on a small number of primitive notions that can often simply be specialized for domain-specific applications. The problems that we address are to establish what the primitives of access control models are and to propose a logic language for their representation. The problem that we consider is not too dissimilar to the one that Landin [25] observed in relation to programming languages: 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). We will argue that by addressing the problem of finding a general model of access control and a logic language for policy formulation, expressed in terms of the model, a number of additional

Special Issue: New Ideas in Applied Logic
Edited by **Dov M. Gabbay and Jacek Malinowski**

Studia Logica (2009) 92: 437–477
DOI: 10.1007/s11225-009-9203-4

problems can be addressed (e.g., the problem of sharing access control policy information).

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 principals, 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). We argue that multiple existing 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. We formalize all of the notions that underlie our approach by developing a new logic language.

In precise detail, the main contributions of the paper are as follows. We introduce a meta-model of access control, we demonstrate that particular access control models are special cases of our meta-model of access control, we formally define the meta-model using a logical framework called Fibred Security Language (FSL) [15, 14] and we show how the generality of FSL allows for the the formal specification of a wide range of access control policies that are particular instances of our meta-model. As such, we define a rich framework for formulating access control policies.

Having a general, unifying access control meta-model provides a basis for a common, well-defined specification of access control requirements. Having a common, agreed semantics is essential when access control information needs to be shared (as is the case with many distributed applications) and is important for reducing the burden on policy administrators when it comes to representing application-specific access control requirements. That is, a policy author can specialize the meta-model to define a domain-specific model and the access control policies that can be represented within that model. These access control policies can be naturally formulated in FSL; thus, a policy author is provided with a general framework for specifying access control requirements. 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).

We also note that an important consequence of our work is to demonstrate 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 and access control languages (cf. [25]).

Although proposals have previously been made for general access control models (see, for example, [22] and [23]), in the ensuing discussion we argue that these approaches are not sufficiently general to warrant the attribution "general access control model" because, as we will see in Section 4, they assume a very limited, particular set of categorizations of principals and have restricted interpretations of some of the key, primitive aspects of access control models.

On the use of logic, in general, for representing access control requirements, we note that logics are important for providing an unambiguous, formal semantics for access control policies, which is essential, for instance, to enable policy authors to prove properties of policy specifications (e.g., security and availability cf. [9]). The particular benefits of using FSL for access control policy specifications include that it provides a very general semantics based on composition of logics, it allows for the specification of a range of modalities that are useful for representing access control requirements in distributed computing contexts, and it allows for the representation of features like joint permission assignment to a group of principals (rather than individual principals). As a drawback, FSL is generally undecidable because it is based on a first order language. We refer to [20] for a discussion on how to constrain first order languages in order to get tractable computational properties in the derivation procedure.

The remainder of this paper is organized thus. In Section 1, 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 2, we present the fibring methodology and we introduce the Fibred Security Language. In Section 3, we describe our meta-model of access control in FSL and we show how a range of existing access control models and some novel access control models may be viewed as instances of our meta-model, when it is formally expressed in terms of FSL, and how FSL may be used as a general language for the specification of access control policies in terms of different access control models (e.g., DAC, RBAC, SBAC). In Section 4, we discuss related work. In Section 5, conclusions are drawn and further work is suggested.

1. Fundamental Concepts

In this section, we describe the conceptual features upon which our work is based.

Our access control meta-model is defined in terms of the primitive notions of categories, relationships between categories and between categories and principals, and modalities.

Intuitively, 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.[1]
- A countable set $\mathcal{A}$ of named *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.

[1] Example of principals are: Users, machines, channels, conjunction of principals, groups ... [4]

Entities in the set $\mathcal{P}$ will include any elements (typically principals 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 a number of access control models).

In addition to the sets of constants above, we include two other sets of constants of special importance in defining the type of general access control framework that we wish to develop:

- A countable set $\mathcal{S}$ of *situational identifiers*.
- A countable set $\mathcal{E}$ of *event identifiers*, e_0, e_1, ...

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.

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 and any number of application-specific relations may be defined in order to satisfy domain-specific requirements. We assume that arithmetic operators $\times, \div, +, -$ and *mod* and comparison operators $=, \geq, \leq, >, <$ and $\neq$ may be used in access control policy specifications.

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, $pca(p, c)$ is 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, $arca(a, r, c)$ is 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 can 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 an assignment of a permission to a specified principal, which can be formalized as follows:

- $\mathcal{PAR}$ 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, $par(p, a, r)$ is 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 as follows:

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

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

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

In concluding this section, we underline a couple of points that should be noted. We have used the word "can" (above) 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. We note too that the RBAC notion of a session [6] can be accommodated in our approach, to wit: a principal can choose to be active in particular categories during a session. However, we view sessions as an operational detail and, as such, this notion is not of concern to us in this paper.

2. Fibring Logics and FSL

In this section, we describe the logic that we use to formalize our meta-model of access control. More specifically, we introduce a *Fibred Security Language (FSL)* [15] for access control in distributed systems. Fibring is a general methodology due to Gabbay [19] that aims at combining logics.

We begin our discussion by noting that first-order logic has proven to be sufficient for representing historically important access control models (e.g., an access control matrix can be viewed as a conjunction of propositions). However, there are access control policy requirements for which classical logic is not sufficient for specification. For example, for the policy information "administrator says that Alice can be trusted when she says to delete $file_1$", Alice speaks for the administrator concerning the deletion of $file_1$ and thus she should be trusted as much as the administrator.

From a semantical point of view, logics for distributed access control rely on one of the following approaches

- Operational Semantics [10].
- Declarative Semantics [13, 26].
- Classical/Intuitionistic Modal logic [1, 2, 4, 24].

Each view has its positive and negative aspects.

Operational Semantics, if rules are wisely crafted, could be extremely clear but very often tractability must be sacrificed for simplicity. SecPAL [10], for instance, has an extremely clear semantics expressed with just three rules, but in practice they are awkward to employ in evaluating formulas. To overcome this difficulty, queries in [10] are evaluated by exploiting Datalog, which has a stable model semantics that is not clearly related with the rules of the operational semantics.

Logics that rely on declarative semantics have a clearly specified notion of proof of compliance of a request with respect to local policy, which is strictly based on the framework in which the reasoning is carried out. On that, PROLOG and Datalog are often used to obtain answer sets from a database of distributed policies. The negative aspect is that when using declarative approaches it could be extremely difficult to have a formal "meaning" for every set of access control policies, such that one can compute this meaning and inspect whether it is the same as the policy author's intention.

Modal logic has been employed by Abadi [4] to model logics for access control. In this view, a logic can be studied through its axiomatization or on the basis of its semantics analyzing how to link models with formulas. One major advantage is that, by making a clear distinction between syntax and semantics, the proof of compliance procedure is based on well-understood, formal foundations. However, a significant disadvantage is that it could be extremely difficult to compose different logics within a common framework if we do not rely on fibring.

Every approach has some positive aspects and each such positive aspect should not be left out in modelling a logic for our meta-model of access control.

To represent the rich forms of access control policies that are often required in distributed applications, we use FSL. On that, suppose that we have two different logics $\mathcal{C}$ and $\mathcal{D}$ with languages $\mathbb{L}_{\mathcal{C}}$, $\mathbb{L}_{\mathcal{D}}$ and semantics $S_{\mathcal{C}}$, $S_{\mathcal{D}}$, respectively. Intuitively, the fibring process consists in defining a combined language $\mathbb{L} \supset \mathbb{L}_{\mathcal{C}} \cup \mathbb{L}_{\mathcal{D}}$ together with a new semantics S in which we can evaluate formulas of both $\mathcal{C}$ and $\mathcal{D}$. This generality will, as we will see, be very important in the context of the meta-model of access control that we will develop and for general access control policy specification for policies that are derivable from the meta-model.

With FSL we propose a general language to compose (by fibring) existing logics on the basis of their semantics; in particular, Section A.2 is devoted to the formalization of an authorization logic called predicate FSL in which we combine by using fibring intuitionistic logic with multimodal logic.

In predicate FSL we have formulas of the kind,

$$\{x\}\varphi(x) \ \textbf{says} \ \psi \tag{I}$$

where $\{x\}\varphi(x)$ represents the group composed by all the principals that satisfy $\varphi(x)$ and ψ is a general formula. We view **says** as a modality for expressing that a certain principal supports some statement (see Section 2.1).

In this view, Formula I becomes

$$\Box_{\{x\}\varphi(x)}\psi \tag{II}$$

In Formula II, ψ is the statement that the extension of $\varphi(x)$ *as a group* of individuals supports; note also that the modality is indexed by principals. To the best of the authors' knowledge, existing approaches that employ the **says** operator do not offer the possibility of having a first-order formula specifying the principals.

Our fibring on access control logics offers freedom in crafting logics to express a wide range of policies. In fact, we can let $\varphi(x)$ and ψ belong to two different languages $\mathbb{L}_p$ and $\mathbb{L}_e$ as language of principals and security expressions, respectively, which refers to two different systems (semantics). For instance, we can think of formulas in $\mathbb{L}_p$ as being SQL queries and formulas in $\mathbb{L}_e$ as being Delegation Logic [26] expressions.

The main problem to be addressed in this context is to specify formally how to evaluate expressions like II; this is the main role of the fibring methodology [19], which, depending on the chosen languages (and systems), must be carefully defined in order to form a combined logic that is coherent and does not collapse.

In this paper, in order to show the full expressiveness of our approach, we decide to make $\mathbb{L}_p = \mathbb{L}_e = \mathbb{L}$, where $\mathbb{L}$ is a classical first order language, whereas the relying system S is intuitionistic modal logic; this is predicate FSL. This particular approach allows us to, for instance, iterate the **says** modality and to have complex formulas in which free variables are shared between different levels of nesting of the $\Box$ (see Section 2.2 for examples).

Moreover, in [3], the intuitionistic semantics has been proven to be more appropriate to axiomatize the **says** modality by designing more expressive logics.

It should be noted that the importance of the FSL framework is that it enables us to define a meta-model of access control that can be specialized to define instances of existing access control models and that can be specialized in novel ways to define instances of new forms of access control models. Moreover, as we will see in Section 3, the FSL framework allows for the representation of any number of specific access control policies which are derivable from the general framework of the meta-model (in terms of which policies are grounded).

2.1. Properties of access control logics

In this section, we first summarize how the **says** operator is used in access control logics and how it is used in our meta-model. We then underline the expressive power of FSL by listing some examples of policy formulation.

The access control logic that we propose aims at distributed scenarios. Thus, to express delegation among principals, it is centered, like the access control logic of [24, 26], on formulas such as "A **says** s" where A represents a principal, s represents a statement (a request, a delegation of authority, or some other utterance), and **says** is a modality. It is important to underline that it is possible to derive that A **says** s even when A does not directly utter s. For example, when the principal A is a user and one of its programs includes s in a message, then we may have A **says** s, if the program has been delegated by A. In this case, A **says** s means that A has caused s to be said, that s has been said on A's behalf, or that A supports s.

We assume that such assertions are used by a reference monitor in charge of making access control decisions for resources, like o (where o denotes an arbitrary data object, e.g., a file). The reference monitor may have the policy that a particular principal A is authorized to perform action a on object o. This policy may be represented by the formula: $(A \textbf{ says } do\_on(a, o)) \rightarrow do\_on(a, o)$, which expresses that A **controls** $do\_on(a, o)$.[2] Similarly, a request for the operation a on o from a principal B may be represented by the formula: B **says** $do\_on(a, o)$. The goal of the reference monitor is to prove that these two formulas imply $do\_on(a, o)$, and to grant access if it succeeds. While proving $do\_on(a, o)$, the reference monitor does not need to prove that the principal B controls $do\_on(a, o)$. Rather it may exploit relations between A and B and certain other facts. For example, it may know that B has been delegated by A, and, thus, that B speaks for A as concerns $do\_on(a, o)$:

$$(B \textbf{ says } do\_on(a, o)) \rightarrow (A \textbf{ says } do\_on(a, o))$$

This simple example does not show the subtleties arising from the formalization of the **says** operator, since expressing simple properties like controlling a resource or speaking for another principal may imply less desirable properties, leading to security risks, or even to inconsistent or degenerate logic systems [3], we refer to [14] for a detailed discussion of these problems with respect to FSL.

[2] In this view. with A **controls** ψ we express that A has a direct permission to do ψ.

The fibring methodology permits us to craft a framework which extends previous logics for access control by introducing joint responsibility between principals and group of principals as first-class citizen described by means of first-order formulas.

2.2. FSL: An extended logic of principals

In this section we make a further step towards predicate FSL by taking into account how to express properties of access control policies in the proposed language, like joint responsibility, delegation or speaks-for relationships.

As underlined in Section 2, in a general FSL formula $\{x\}\varphi(x)$ **says** ψ we use $\{x\}\varphi(x)$ as a construct to select the set of principals making the assertion **says**. Note that $\varphi(x)$ and ψ can share variables and φ may include occurrences of the **says** operator. Notice too that x can occur in ψ but then this occurrence is not related to the x in $\{x\}\varphi(x)$.

To select a single principal whose name is A we write:

$$\{x\}(x = A) \textbf{ says } \psi.$$

We write A **says** ψ **s** for $\{x\}(x = A)$ **says** ψ, where A is an individual principal.

The following formula means that all $user$s together ask to delete $file_1$:

$$\{x\}user(x) \textbf{ says } do\_on(delete, file_1)$$

Since $\varphi(x)$ and ψ can share variables, we can put restrictions on the variables occurring in ψ. For example, the set of all users who all own file(s) y asks to delete the file(s) y:

$$\{x\}(user(x) \wedge own(x, y)) \textbf{ says } do\_on(delete, y)$$

However, the formula above is satisfactory only in the particular situation where we are talking about the set of all users who assert **says** at once as a group (committee).

We can also express that each member of a set identified by a formula can assert **says** separately. For example, the policy requirement "each user deletes individually the files he owns", can be represented thus:

$$\forall x(user(x) \wedge own(x, y)) \rightarrow \{z\}(z = x) \textbf{ says } delete(y).$$

Note that the latter formula usually implies the former but not vice versa.[3]

More generally, in FSL you have the possibility to express how the set $\{x \mid \varphi(x) holds\}$ says what it says. For instance, suppose we have $\varphi(x) \equiv (x = A_1) \vee (x = A_2) \vee (x = A_3)$ then if at least one of $\{A_i\}$ **says** ψ is enough for the group to support ψ we add:

$$\{x\}\varphi(x) \textbf{ says } \psi \leftrightarrow \bigvee_{1 \leq i \leq 3} \{x\}(x = A_i) \textbf{ says } \psi$$

This represents the fact that each principal in the group can speak for the whole group. We can as well express that group φ has a spokes-person y:

$$\textbf{spoke}(\varphi, y) = (\forall X[\{x\}\varphi(x) \textbf{ says } X \leftrightarrow \{x\}(x = y) \textbf{ says } X])$$

In FSL, with features like the sharing of variables between $\varphi(x)$ and ψ, the nesting of the says and the employment of negation we can express complex policies like separation of duties in a compact way. For instance, we can express the following: "A member m of the Program Committee can not accept a paper P_1 in which one of its authors says that he has published a paper with him after 2007"

$$\neg(\{m\}[PC(m) \wedge \{y\}author\_of(y, P_1) \textbf{ says } \exists p(paper(p) \wedge author\_of(m, p) \wedge author\_of(y, p) \wedge year(p) \geq 2007)] \textbf{ says } accept(P_1))$$

For a deeper treatment of the expressive power of the language we refer to [14].

3. An Access Control Meta-Model in Predicate FSL

In this section, we define our meta-model of access control in terms of predicate FSL, as the latter has been defined in the previous section.

Our meta-model of access control, henceforth denoted by $\mathcal{M}$, is based on a single core axiom, which is derived from the first order expression of *par* from Section 1 and which we report here:

[3] In fact, it could be sensible to have situations in which if all the members of a group say something then the whole group says it but not conversely:

$$\forall x(\varphi(x) \rightarrow x \textbf{ says } \psi) \rightarrow \{x\}\varphi(x) \textbf{ says } \psi$$

For instance, a committee may approve a paper that not all of its members would have accepted.

$$pca(P, C) \wedge \rho(C, C') \wedge arca(A, R, C') \rightarrow par(P, A, R)$$

By choosing different definitions of *pca*, ρ 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 ρ condition if that is not required for a domain-specific application, and it is equally possible for more than one ρ relation to be defined in an instance of $\mathcal{M}$.

The integration of the meta-model, first introduced in [7], with predicate FSL involves the following mapping:

- **Categories as types**: Instead of having the *pca* relationship we view categories as first-order formulas with one free variable $C(x)$. We say that a principal p is assigned to category C (expressed by a first order formula) iff $\models_{\mathcal{A}} C(p)$ holds in the first order structure $\mathcal{A}$. So, intuitively, we translate *pca* as follows:

  $$pca(P, C) \equiv C(P).$$

  In fact, in FSL, group membership is expressed by means of first order formulas. Moreover, by having categories as types we can express roles as a special instance of categories, as in RT [27] (see Section 3.2 for a detailed discussion).

- ρ **relationship as operator**: If we view categories as types, we have to redefine the ρ relationship. We do this formally by considering two different levels:
  - First-order structure $\mathcal{A}$: We define the meta-model relationship $\rho^{\mathcal{A}}$ as a subset of $2^P \times 2^P$.[4]
  - Meta-model merged with FSL: We define a novel binary operator ρ with the following semantics:

    $$\models_{\mathcal{A}} \rho(\varphi(x), \psi(y)) \text{ iff } (\{p \mid\models_{\mathcal{A}} \varphi(p)\}, \{t \mid\models_{\mathcal{A}} \psi(t)\}) \in \rho^{\mathcal{A}}$$

- *par* **and** *arca* **relationships**: Both *par* and *arca* refer to permission assignment, the only difference between them is the fact that *par* refers to a single principal, whereas *arca* refers to single category. If we view categories as types, we have that *arca* refers to *sets* of principals, so we can exploit the FSL machinery to apply the following translation:

  $$par(P, A, R) \equiv P \textbf{ controls } do\_on(A, R)$$
  $$arca(A, R, C(x)) \equiv C(x) \textbf{ controls } do\_on(A, R)$$

[4] That is, as a relationship between sets of principals.

This view offers the possibility to have a more fine-grained definition of how *arca* is related with *par*, for instance:

- If a category $c(x)$ has the permission to do action a on r, then all the principals assigned to that category have the same permission:

$$(c(x) \textbf{ controls } do\_on(a, r)) \rightarrow \forall t(c(t) \rightarrow t \textbf{ controls } do\_on(a, r))$$

- If a category $c(x)$ has the permission to do action a on r, then a spokes-person for the group, identified by $c(x)$, has the direct control of doing a on r:

$$(c(x) \textbf{ controls } do\_on(a, r)) \rightarrow (\textbf{spoke}(c(x), t) \rightarrow t \textbf{ controls } do\_on(a, r))$$

In the next three sections, we consider alternative definitions of the predicates *pca*, ρ and *arca* within the FSL framework and the different access control models that can be naturally derived from $\mathcal{M}$ by changing the definition.

3.1. *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 3.1. Definitions of *pca* are expressed in the form:

$$\mathcal{P}_1 \wedge \cdots \wedge \mathcal{P}_n \wedge L_1 \wedge \cdots \wedge L_p \wedge C_1 \wedge \cdots \wedge C_m \rightarrow C(P)$$

$\mathcal{P}_i$ $(1 \leq i \leq n)$ 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 that are expressed in terms of the arithmetic or comparison operators that we admit in our language.

Once a category $C(x)$ has been defined by a policy author α, α's definition of $C(x)$ 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 C 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):

$$loyal(P) \wedge goodbalance(P) \rightarrow preferred(P)$$
$$balance(P, X) \wedge X \geq 1000 \rightarrow goodbalance(P)$$

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 time. 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, [9]), but these are just particular 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 *trusted_on_uni* category and Y says that P is assigned to the category *good_university*.

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

$$trusted\_on\_uni(Y) \wedge \Box_Y(good\_university(P)) \rightarrow approved\_uni(P) \qquad \Box$$

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

$$manager\_b_1(P)$$

More generally, variables may be used in paramaterized expressions.

3.2. ρ 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 relations between categories (ρ) 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 [17]. 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 [6, 18] 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 as follows (where '_' denotes an anonymous variable):

$$C(P) \wedge \rho(C(x), C'(y)) \wedge$$
$$c'(y) \textbf{ controls } do\_on(a, r) \rightarrow P \textbf{ controls } do\_on(a, r)$$
$$dc(C, \_) \rightarrow \rho(C, C)$$
$$dc(\_, C) \rightarrow \rho(C, C)$$
$$dc(C', C'') \rightarrow \rho(C', C'')$$
$$dc(C', C''') \wedge \rho(C''', C'') \rightarrow \rho(C', C'')$$

In this instance, ρ is a definition of a partial order relationship between pairs of categories (here restricted to roles) that are in the reflexive-transitive 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 theory 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 ρ 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)* [8].

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 ρ assumed and with T - T_s being the interval of time during which a relationship holds):[5]

$$C(P) \wedge \rho(C(x), C'(y)) \wedge$$
$$C'(y) \textbf{ controls } do\_on(A, R) \rightarrow P \textbf{ controls } do\_on(A, R)$$

$$current\_time(T) \wedge happens(E, T_s) \wedge$$
$$agent(E, P) \wedge act(E, A) \wedge T_s < T \wedge$$
$$pca\_init(E, P, A, C, T_s, T) \wedge$$
$$not\ ended\_pca(P, C, T_s, T) \rightarrow C(P)$$

$$happens(E', T') \wedge agent(E', P) \wedge$$
$$act(E', A') \wedge pca\_term(E', P, A', C, T_s, T) \wedge$$
$$T_s < T' \wedge T' \leq T \rightarrow ended\_pca(P, C, T_s, T)$$

By changing the definitions of *pca* and ρ (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 ρ 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., $dc(europe, uk)$). In this case, *pca* may be defined by using rules of the standard form,

$$\mathcal{P}_1 \wedge \cdots \wedge \mathcal{P}_i \wedge L_1 \wedge \cdots \wedge L_m \wedge C_1 \wedge \cdots \wedge C_n \rightarrow C'(P)$$

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

[5] For the definitions of SBAC-specific predicates, like *pca_init*, we refer the reader to [8].

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, [30]). In terms of our proposed framework, a version of the Bell-LaPadula model [11] may be viewed as a restricted form of the Hierarchical RBAC model, in which ρ is as previously defined and with *par* defined thus:

$$C(P) \wedge \rho(C(x), C'(y)) \wedge$$
$$C'(y) \textbf{ controls } do\_on(R, C) \rightarrow P \textbf{ controls } do\_on(read, R)$$

$$C(P) \wedge \rho(C(x), C'(y)) \wedge$$
$$C(x) \textbf{ controls } do\_on(write, R) \rightarrow P \textbf{ controls } do\_on(write, R)$$

In this case, the ρ 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 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 ρ 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'. In this case, members of c' are in the category of being delegatees of the delegator c. From the logical point of view, delegation can be represented by the speaks-for relationship 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:

$$current\_time(T) \wedge C_{[T',T'']}(P) \wedge \rho(C'(x), C''(y)) \wedge$$
$$C(x) \textbf{ controls}_{[T''',T'''']}\ do\_on(A, R) \wedge$$
$$T' \leq T \wedge T \leq T'' \wedge T''' \leq T \wedge T \leq T'''' \rightarrow P \textbf{ controls } do\_on(A, R)$$

Notions like relative times and periodic times (cf. [12]) 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:

$$\Box_{\upsilon_1} P \textbf{ controls } do\_on(A, R), \Box_{\upsilon_2} P \textbf{ controls } do\_on(A, R) \rightarrow$$
$$P \textbf{ controls } do\_on(A, R)$$

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 (TTPs) in our framework by allowing specifications of properties by $\Box_\upsilon p(\tau)$ (i.e. υ **says** $p(\tau)$) and relations by $\Box_\upsilon p(\tau_1, \ldots, \tau_n)$ (where τ and $\tau_i, i \in \{1, \ldots, n\}$ are terms and υ is the source of the TTP assertion). We note that, interpreted in terms of certification-based access control, our use of relations of the form $\Box_\upsilon p(\tau_1, \ldots, \tau_n)$ 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 a certificate space υ (cf. SPKI certificates [16]).

On the *RT* family of role-trust access control models specifically, we note that the proposers of the *RT* family [27] 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 $\mathcal{M}$ 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 $\tau_1, \ldots, \tau_n$, $\sigma_1, \ldots, \sigma_m$

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

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):

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

Other forms of credentials 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 3. 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 v_2) of any principal that is assigned to the category of manager ((mgr) details recorded locally) of a branch that is categorized as profitable (as recorded in v_3). To be categorized as a senior executive, a principal must be categorized as being a manager of at least five years standing (according to the source of this information, v_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 a policy specification expressed in terms of predicate FSL (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):

$$C(P) \wedge \rho(C(x) \wedge C'(y)) \wedge$$
$$C'(y) \textbf{ controls } do\_on(A, R) \rightarrow P \textbf{ controls } do\_on(A, R)$$

$$\Box_{v_1} mgr_T(P) \wedge current\_time(T') \wedge year(T, Y) \wedge year(T', Y') \wedge$$
$$Y' - Y \geq 5 \rightarrow s\_exec(P)$$

$$manager_Y(X) \wedge \Box_{v_3} profitable(Y) \wedge \Box_{v_2} salary(X, Y) \rightarrow$$
$$s\_exec(x) \textbf{ controls } do\_on(read, salary(X, Y)) \qquad \Box$$

Any number of additional, novel forms of access control models may be similarly defined in terms of $\mathcal{M}$ and then expressed in terms of FSL. For instance, suppose that an access control model were required with a type of ρ 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 [28]). 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 FSL and in terms of *dc*, a ρ relation of the following form:[6]

$$dc(C, C') \wedge dc(C, C'') \rightarrow \rho(C', C'')$$

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" [31]). Thus, 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 use FSL to define a rule of the following form:

$$\rho(C', C) \rightarrow \rho(C, C')$$

[6] More complex forms of this type of Euclidean relation are, of course, clearly possible.

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 $\mathcal{M}$ and that policies that are specified in terms of $\mathcal{M}$ can be naturally represented in predicate FSL. Moreover, multiple "novel" forms of existing access control models may be similarly developed from the core axioms and predicates of $\mathcal{M}$ and can be expressed in policy specifications using FSL (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. These constraints have a natural representation in terms of predicate FSL.

In Constrained RBAC [18], 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 $\mathcal{M}$ and FSL, in terms of *pca*, thus (where $\perp$ read as "is inconsistent" and $c \in \mathcal{C}$ and $c' \in \mathcal{C}$ are constants that denote specific categories):

$$C(P) \wedge C'(P) \rightarrow \perp$$

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

$$C(P) \wedge C'(P) \wedge me(C, C') \rightarrow \perp$$

can be used to define arbitrary pairs of categories that are mutually exclusive i.e., *me* is the case. 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 $\mathcal{M}$ and expressed using predicate FSL. For example, the constraint,

$$C(P) \wedge \neg C'(P) \rightarrow \perp$$

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 $\mathcal{M}$ and can be expressed using predicate FSL. 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 $\mathcal{M}$ and can be expressed using predicate FSL in order to specify that no principal with the status of *debtor* (as recorded in the TTP source, v_1) can be cleared to access anything other than the resources that are accessible to principals with unclassified clearance (as recorded in the TTP source, v_2):

$$\Box_{v_1} debtor(P) \wedge \Box_{v_2} C(P) \wedge C \neq unclassified \rightarrow \perp$$

Constraints may be similarly expressed, using FSL, 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 constructs of $\mathcal{M}$ and FSL 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 r_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 $\mathcal{M}$ and expressed using FSL:

$$\begin{aligned} &happens(E,T) \wedge actor(E,P) \wedge action(E,read) \wedge resource(E,r_1) \wedge \\ &happens(E',T') \wedge actor(E',P) \wedge action(E',read) \wedge \\ &resource(E',r_1) \wedge E \neq E' \wedge T' - T < 1 \rightarrow \perp \end{aligned}$$

In general, by combining the elements of $\mathcal{M}$ and by using FSL for policy formulation, any number of specific access control models and access control policies may be defined in using the general framework that we have introduced.

3.3. *arca* definitions

Thus far, our discussion has been focused on demonstrating how predicate FSL may be used to express a wide range of access control models and policies that can be expressed in terms of our meta-model $\mathcal{M}$, by changing the definitions of *pca* and ρ, and *par*. However, there is, as we have mentioned previously, a useful generalization of the notion of permissions that is very 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. Moreover, predicate FSL provides a means for specifying access control policy requirements in this context.

To motivate the discussion on this point, consider the language of the *Flexible Authorization Framework (FAF)* [22]. 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. These rules are defined in essentially the same way as *pca* rules are defined.

Definition 3.2. Rules defining *arca* are expressed in the following general form:

$$\mathcal{A}_1 \wedge \cdots \wedge \mathcal{A}_n \wedge L_1 \wedge \cdots \wedge L_p \wedge C_1 \wedge \cdots \wedge C_m \rightarrow arca(A, R, C)$$

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 latter case, the condition is of the form $\Box_\upsilon A_i$ or $\Box_\upsilon L_i$ 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 4. 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 policy requirements in terms of $\mathcal{M}$ and predicate FSL, the following rules may be used:

$$\Box_{\upsilon_1} stock(gold, Y) \wedge Y - X \geq 0 \wedge type(C) \neq debtor \wedge X \leq 50 \wedge$$
$$\neg\Box_{\upsilon_2} market(gold, volatile) \rightarrow C(x) \textbf{ controls } do\_on(buy, gold(X))$$

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 FSL specification 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 FSL specification 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 in the FSL specification emphasize that different aspects of "can" need to be separately specified.

Any number of constraints may be expressed in terms of *arca*. For example, in predicate FSL,

$$C(x) \textbf{ controls } do\_on(write, o_1) \wedge \neg(C'(x) \textbf{ controls } do\_on(write, o_1)) \rightarrow \bot$$

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, in predicate FSL,

$$arca_c(A, R, C) \wedge not\ arca_p(A, R, C) \rightarrow \bot$$

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 $\mathcal{M}$ and these denials can be expressed in predicate FSL. It should be noted that a *d_arca* predicate (say) could be used in FSL specifications 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.).

4. Related Work

In the discussion above, we have described a very general framework for the specification of access control models and policies that is based on a meta-model of access control and a fibred logic for policy specification.

As we previously mentioned, other researchers have also attempted to define general frameworks for access control for example, the Generalized TRBAC model and ASL [23, 22]. 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 should be clear that, despite the extensive literature on RBAC, it is our view that RBAC is 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 ρ 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 shareability of access control policy information. More importantly, there are policies that simply cannot be expressed in RBAC because of its restricted interpretation. For example, RBAC only admits a single form of ascribed status (cf. [8]) that is based on a particular category (the role) interpreted in functional terms (cf. the notion of a "job function"). As such, RBAC does not allow for action status to be captured, for example. Moreover,

permission assignment in RBAC does not allow for notions to be expressed like "can do" as physical capability. 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 $\mathcal{M}$.

We also note that a general *language* for access control policy specification has already been described in the access control literature: XACML [29]. 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 without a generally accepted formal semantics, as is the case with XACML). Unlike XACML, predicate FSL is based on a well defined formal semantics and allows for rich, declarative specifications of access control policies. In addition to its unsatisfactory formal underpinnings, XACML is not based on a well defined conceptual model of access control. In contrast, in our approach the meta-model $\mathcal{M}$ is well defined, conceptually and in terms of FSL.

A sceptical reader might 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, it is our view that 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 [25]). Our meta-model and our use of predicate FSL does provide a general framework for representing a wide range of access control concepts in a uniform manner.

Some may also argue that any access control model that is claimed to be general, as RBAC has been suggested to be [9], 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 argued 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.

In relation to our use of predicate FSL for representing access control requirements, the work by Abadi et al. [5] exhibits similarities to ours. As with predicate FSL, ABLP logic provides a formal framework for reasoning about features of access control. A number of connectives are included in the logic for representing access control requirements (e.g., P *as* R for specifying principal P in role R). However, the focus in ABLP logic is on language constructs for formulating access control policies and axioms and

inference rules for defining a system for proof (e.g., for proving authorized forms of access). In contrast, our approach is based on deriving, from the generalities of access control models, the common aspects of access control from which core predicates are identified and given a fixed interpretation in predicate FSL; application-specific predicates are added to define instances of the meta-model.

Our approach is also related to those adopted by Barker and Stuckey [9] and Jajodia et al. [22]. In these approaches, predicates with fixed interpretations are identified for access control specification in a logic language. However, in both of these approaches the logic languages that are employed are less expressive than predicate FSL, they are based on specific access control models (role-based and discretionary models) and neither approach captures the generalities of access control that are the basis for our meta-model. Moreover, both approaches assume that a centralized system is to be protected. Our meta-model and our fibred logic provide a more general access control framework.

The work by Li et al. [27] 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 (credentials) are included in specific elements of the family of *RT* models. In contrast, we allow rules for defining *pca*, *arca* and *par* in predicate FSL, in general. As such, in our meta-model it is possible to specify access control requirements in terms of the non-assignment of principals to a category, for example.

The work on SecPAL [10] is motivated, as ours is, by the esire 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, expressed in predicate FSL, are derived directly from the meta-model.

5. Conclusions and Further Work

In this paper, we have considered the key question of how best to address the problem of representing complex access control requirements in the context of centralized and distributed computing. To answer this question, we have proposed a novel methodological approach that focuses on the generalities of access control rather than particulars. For that, we defined a meta-model of access control that is based on the notion of a category; the notion of category generalizes the particular categories that are used in access control, e.g., role, status, classification, clearance, ... The meta-model includes three core, primitive predicates for expressing principal-category assignments (pca), permission-category assignments ($arca$) and category-category relationships. We also illustrated how a range of constraints may be represented in terms of our meta-model and how a rich interpretation of permission assignment can be accommodated to allow for different interpretations of "can do". We illustrated how this framework for access control policy formulation allows a number of existing access control models to be represented and how a number of "novel" access control models may be developed by specializing the meta-model. For policy formulation in relation to our meta-model of access control, we have developed a logic language.

We have argued that having a rich framework, from which specific access control models, policies and policy specification languages may be derived, has an number of attractions. For example, the framework that we have described facilitates the sharing of policy information (e.g. by policy composition). Having a rich logic language in terms of which our meta-model can be defined also makes it possible to have a shared semantics for different access control models and makes it possible to prove properties of access control policies that are defined in terms of the meta-model. Moreover, providing a general framework of access control and an expressive policy language abstracts away many of the complexities that are involved in policy representation in computing environments (centralized and decentralized) and simplifies the task of policy authors that are required to capture policy requirements.

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 $\mathcal{M}$ and predicate FSL. 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 $\mathcal{M}$ and their formulation using FSL are also matters for further work.

Appendix

A. The basic system FSL

In this section we introduce our basic system FSL step-by-step from a semantic viewpoint[7]. For that, we first introduce modalities indexed by propositional atoms, then we take into account classical and intuitionistic models for the propositional setting, and finally we give a semantics to predicate FSL, which we have extensively employed in previous sections of this paper.

This system can be defined with any logic $\mathbb{L}$ as a *Fibred Security System based on* $\mathbb{L}$. We will motivate the language for the cases of $\mathbb{L}$ = classical logic and $\mathbb{L}$ = intuitionistic logic.

Basically, adding the **says** connective to a system is like adding many modalities. So to explain and motivate FSL technically we need to begin with examining options for adding modalities to $\mathbb{L}$. Section A.1 examines our options of how to add modalities to classical and intuitionistic logics. The presentation and discussion is geared towards section A.2, which presents predicate FSL.

A.1. Adding modalities

We start by adding modalities to classical propositional logic; our approach is semantic.

Let S be a nonempty set of possible worlds. For every subset $U \subseteq S$ consider a binary relation $R_U \subseteq S \times S$.

This defines a multi-modal logic, containing K modalities $\Box_U$, $U \subseteq S$. The models are of the form $(S, R_U, t_0, h), U \subseteq S$. In this view, if $U = \{t | t \vDash \varphi_U\}$ for some φ_U then we get a modal logic with modalities indexed by formulas of itself.

[7]This section is extensively based on Section 4 of [15], we decided to report it here to make the article self-contained.

Definition A.1 (Language). Consider (classical or intuitionistic) propositional logic with the connectives $\wedge, \vee, \rightarrow, \neg$ and a binary connective $\Box_\varphi \psi$, where φ and ψ are formulas. The usual definition of a wff is adopted.

Definition A.2. We define classical Kripke models for this language.

1. A model has the form

$$\mathbf{m} = (S, R_U, t_0, h), U \subseteq S$$

   where for each $U \subseteq S, R_U$ is a binary relation on S, $t_0 \in S$ is the actual world and h is an assignment, giving for each atomic q a subset $h(q) \subseteq S$.

2. We can extend h to all formulas by structural induction:
   - $h(q)$ is already defined, for q atomic
   - $h(A \wedge B) = h(A) \cap h(B)$
   - $h(\neg A) = S - h(A)$
   - $h(A \rightarrow B) = (S - h(A)) \cup h(B)$
   - $h(A \vee B) = h(A) \cup h(B)$
   - $h(\Box_\varphi \psi) = \{t |$ for all s $(tR_{h(\varphi)}s \rightarrow s \in h(\psi))\}$
3. $\mathbf{m} \vDash A$ iff $t_0 \in h(A)$.

Notice that there is nothing that is particularly new about this except possibly the way we are looking at it.

Let us now do the same for intuitionistic logic. Here things become more interesting. An intuitionistic Kripke model has the form,

$$\mathbf{m} = (S, \leq, t_0, h)$$

where $(S, \leq)$ is a partially ordered set, $t_0 \in S$ and h is an assignment to the atoms such that $h(q) \subseteq S$. We require that $h(q)$ is a closed set, that is:

$$x \in h(q) \text{ and } x \leq y \text{ imply } y \in h(q).$$

Let D be a set; we can add for each $U \subseteq D$ a binary relation R_U on S. This semantically defines an intuitionistic modality, $\Box_U$.

In intuitionistic models we require the following condition to hold for each formula A, i.e. we want $h(A)$ to be closed:

$$x \in h(A) \text{ and } x \leq y \Rightarrow y \in h(A).$$

This condition holds for A atomic and propagates over the intuitionistic connectives $\wedge, \vee, \rightarrow, \neg, \bot$. To ensure that it propagates over $\Box_U$ as well, we need an additional condition on R_U. To see what this condition is supposed to be, assume $t \vDash \Box_U A$. This means that

$$\forall y(tR_U y \Rightarrow y \vDash A).$$

Let $t \leq s$. If $s \nvDash \Box_U A$, then for some z such that $sR_U z$ we have $z \nvDash A$. This situation is impossible if we require

$$t \leq s \wedge sR_U z \Rightarrow tR_U z. \qquad (*)$$

Put differently, if we use the notation:

$$R'_U(x) = \{y | xR_U y\}$$

then

$$x \leq x' \Rightarrow R'_U(x) \supset R'_U(x'). \qquad (*)$$

So we now talk about modalities R_U, for $U \subseteq S$. We ask what happens if U is defined by a formula φ_U, i.e. $U = h(\varphi_U)$. This will work only if U is closed

$$t \in U \wedge t \leq s \Rightarrow s \in U.$$

Henceforth, we talk about modalities associated with closed subsets of S.

We can now define our language (cf. Definition A.1). We first define the semantics.

Definition A.3. A model has the form,

$$\mathbf{m} = (S, \leq, R_U, t_0, h), U \subseteq S,$$

where $(S, \leq)$ is a partial order, $t_0 \in S$, and each $U \subseteq S$ is a closed set and so is $h(q)$ for atomic q. R_U satisfies condition (*) above. We define the notion $t \vDash A$ for a wff by induction, and then define

$$h(A) = \{t | t \vDash A\}.$$

Next, we define $\vDash$:

- $t \vDash q$ iff $t \in h(q)$
- $t \vDash A \wedge B$ iff $t \vDash A$ and $t \vDash B$
- $t \vDash A \vee B$ iff $t \vDash A$ or $t \vDash B$

- $t \vDash A \to B$ iff for all $s, t \leq s$ and $s \vDash A$ imply $s \vDash B$
- $t \vDash \neg A$ iff for all s, $t \leq s$ implies $s \nvDash A$
- $t \nvDash \bot$
- $t \vDash \Box_{\varphi}\psi$ iff for all s such that $tR_{h(\varphi)}s$ we have $s \vDash \psi$. We assume by induction that $h(\varphi)$ is known.
- $\mathbf{m} \vDash A$ iff $t_0 \vDash A$.

It is our intention to read $\Box_{\varphi}\psi$ as φ **says** ψ.

A.2. Predicate FSL

Intuitively, a predicate FSL model is represented by a set of models linked together by means of a *fibring function*; every model has an associated domain D of elements together with a set of formulas that are true in it. In the FSL model, the evaluation of the generic formula $\{x\}\varphi(x)$ **says** ψ is carried out in two steps, first evaluating φ and then ψ in two different models. Suppose $\mathbf{m_1}$ is our (first order) starting model in which we identify $U \subseteq D$ as the set of all the elements that satisfy φ. Once we have U we can access one or more worlds depending on the *fibring function* $\mathbf{f} : \mathcal{P}(D) \to \mathcal{P}(M)$, which goes from sets of elements in domain D to sets of models. At this point, for every model $\mathbf{m_i} \in \mathbf{f}(U)$ we must check that ψ is *true*; if this is the case then α is true in the meta-model.

The fact that in the same expression we evaluate different sub-formulas in different models it is not completely counterintuitive. For instance, consider a group of administrators that have to set up security policies for their company. From a semantical point of view, if we want to check if ψ holds in the depicted configuration by the administrators, we must

1. Identify all the administrators (all the elements that satisfy $admin(x)$).
2. Access the model that all the administrators as a group have depicted.
3. Check in that model if ψ is *true* or *false*

Let $\mathbb{L}$ denote classical or intuitionistic predicate logic.[8] We assume the usual notions of variables, predicates, connectives $\wedge, \vee, \to, \neg$, quantifiers $\forall, \exists$ and the notions of free and bound variables.

Let $\mathbb{L}^+$ be $\mathbb{L}$ together with two special symbols:

- A binary (modality), x **says** y.

[8]Classical predicate logic and intuitionistic predicate logic have the same language. The difference is in the proof theory and in the semantics.

- A set-binding operator $\{x\}\varphi(x)$ meaning the set of all x such that $\varphi(x)$.

Note that semantically, in the appropriate context, $\{x\}\varphi(x)$ can behave like $\forall x\varphi(x)$ and sometimes in other contexts, we will use it as a set.

Definition A.4. The language FSL has the following expressions:

1. All formulas of $\mathbf{L}^+$ are level 0 formulas of FSL.
2. If $\varphi(x)$ and ψ are formulas of $\mathbb{L}^+$ then $\alpha = \{x\}\varphi(x)$ **says** ψ are level 1 'atomic' formulas of FSL. If $(x, x_1, \ldots, x_n)$ are free in φ and $y_1, \ldots, y_m$ are free in ψ then $\{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ are free in α. The variable x in φ gets bound by $\{x\}$. The formula of level 1 are obtained by closure under the connectives and quantifiers of $\mathbb{L}^+$.
3. Let $\varphi(x)$ and ψ be formulas of FSL of levels r_1 and r_2 resp., then $\alpha = \{x\}\varphi$ **says** ψ is an 'atomic' formula of FSL of level $r = \max(r_1, r_2) + 1$.
4. Formulas of level n are closed under classical logic connectives and quantifiers of all 'atoms' of level $m \leq n$.

Definition A.5 (FSL classical fibred model of level n).

1. Any classical model with domain D is an FSL model of level 0.
2. Let $\mathbf{m}$ be a classical model of level 0 with domain D and let for each subset $U \subseteq D, \mathbf{f}^n(U)$ be a family of models of level n (with domain D). Then $(\mathbf{m}, \mathbf{f}^n)$ is a model of level $n + 1$.

Definition A.6 (Classical satisfaction for FSL). We define satisfaction of formulas of level n in classical models of level $n' \geq n$ as follows.

First observe that any formula of level n is built up from atomic predicates of level 0 as well as 'atomic' formulas of the form $\alpha = \{x\}\varphi(x)$ **says** ψ, where φ and ψ are of lower level.

We therefore first have to say how we evaluate $(\mathbf{m}, \mathbf{f}^n) \vDash \alpha$.

We assume by induction that we know how to check satisfaction in $\mathbf{m}$ of any $\varphi(x)$, which is of level $\leq n$.

We can therefore identify the set $U = \{d \in D \mid \mathbf{m} \vDash \varphi(d)\}$.

Let $\mathbf{m}' \in \mathbf{f}^n(U)$. We can now evaluate $\mathbf{m}' \vDash \psi$, since ψ is of level $\leq n-1$.

So we say

$$(\mathbf{m}, \mathbf{f}^n) \vDash \alpha \text{ iff for all } \mathbf{m}' \in \mathbf{f}^n(U), \text{ we have } \mathbf{m}' \vDash \psi.$$

We need to add that if we encounter the need to evaluate $\mathbf{m} \vDash \{x\}\beta(x)$, then we regard $\{x\}\beta(x)$ as $\forall x\beta(x)$.

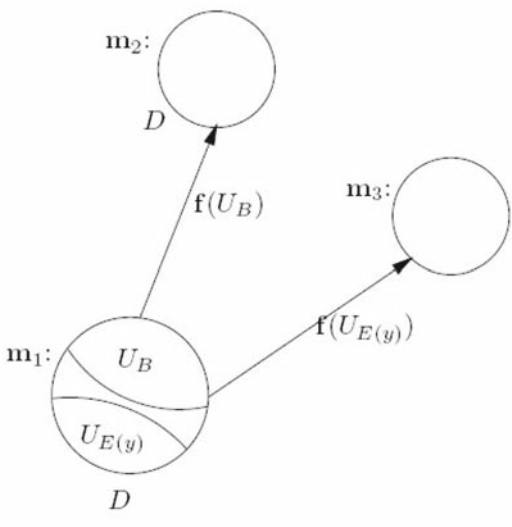

Figure 1.

Example 3. Figure 1 is a model for

$$\alpha(y) = \{x\}[\{u\}B(u) \textbf{ says } (B(x) \to A(x,y))] \textbf{ says } F(y).$$

In Figure 1, $\mathbf{m}_1$ is a single model in $\mathbf{f}^1(U_B)$ and $\mathbf{m}_3$ is a single model in $\mathbf{f}^1(U_{E(y)})$, as defined later. □

The set U_B is the extension of $\{x\}B(x)$ in $\mathbf{m}_1$.

To calculate the set of pairs (x,y) such that $E(x,y) = \{u\}B(u)$ **says** $(B(x) \to A(x,y))$ holds in $\mathbf{m}_1$, we need to go to $\mathbf{m}_2$ in $\mathbf{f}(U_B)$ and check whether $B(x) \to A(x,y)$ holds in $\mathbf{m}_2$, x, y are free variables so we check the value under fixed assignment.

We now look at $E(y) = \{x\}E(x,y)$ for y fixed, we collect all elements d in D such that $\mathbf{m}_2 \vDash B(d) \to A(d,y)$. Call this set $U_{E(y)}$.

To check $\alpha(y) = \{x\}E(x,y)$ **says** $F(y)$ in $\mathbf{m}_1$ we have to check whether $F(y)$ holds in $\mathbf{m}_3$.

We now define intuitionistic models for predicate FSL. This will give a semantics for the intuitionistic language.

Definition A.7. We start with intuitionistic Kripke models which we assume for simplicity have a constant domain. The model $\mathbf{m}$ has the form $(S, \leq, t_0, h, D)$ where D is the domain and $(S, \leq, t_0)$ is a partial order with

first point t_0 and h is an assignment function giving for each $t \in S$ and each m-place atomic predicate P a subset $h(t, P) \subseteq D^m$ such that $t_1 \leq t_2 \Rightarrow h(t_1, P) \subseteq h(t_2, P)$

We let $h(P)$ denote the function $\lambda t\ h(t, P)$. For $t \in S$ let

$$S_t = \{s \mid t \leq s\}$$
$$h(t, P) = h(P) \restriction S_t$$
$$\leq_t = \leq \restriction S_t$$

Where $\restriction$ is the classical restriction operator.

Let $\mathbf{m}_t = (S_t, \leq_t, t, h_t, D)$.

Note that a formula φ holds at $\mathbf{m} = (S, \leq, t_0, h, D)$ iff $t_0 \vDash \varphi$ according to the usual Kripke model definition of satisfaction.

1. A model of level 0 is any model $\mathbf{m}$: $\mathbf{m} = (S, \leq, t_0, h, D)$.
2. Suppose we have defined the notion of models of level $m \leq n$, (based on the domain D).

We now define the notion of a model of level $n + 1$

Let $\mathbf{m}$ be a model of level 0 with domain D. We need to consider not only $\mathbf{m}$ but also all the models $\mathbf{m}_t = (S_t, \leq_t, t, h_t, D)$, for $t \in S$. The definitions are given simultaneously for all of them.

By an intuitionistic 'subset' of D in $(S, \leq, t_0, h, D)$, we mean a function $\mathbf{d}$ giving for each $t \in S$, a subset $\mathbf{d}(t) \subseteq D$ such that $t_1 \leq t_2 \Rightarrow \mathbf{d}(t_1) \subseteq \mathbf{d}(t_2)$.

Let $\mathbf{f}_t^n$ be a function associating with each $\mathbf{d}_t$ and $t \in S$ a family $\mathbf{f}_t^n(\mathbf{d}_t)$ of level n models, such that $t_1 \leq t_2 \Rightarrow \mathbf{f}_{t_1}^n(\mathbf{d}_{t_1}) \supseteq \mathbf{f}_{t_2}^n(\mathbf{d}_{t_2})$. Then $(\mathbf{m}_t, \mathbf{f}_t)$ is a model of level $n + 1$ where $\mathbf{d}_t = \mathbf{d} \restriction S_t$.

Definition A.8 (Satisfaction in fibred intuitionistic models). We define satisfaction of formulas of level n in models of level $n' \geq n$ as follows.

Let $(\mathbf{m}_t, \mathbf{f}_t^n)$ be a level n model. Let $\alpha = \{x\}\varphi(x)$ **says** ψ is of level n. We assume we know how to check satisfaction of $\varphi(x)$ in any of these models.

We can assume that

$$\mathbf{d}_t = \{x \in D \mid t \vDash \varphi(x) \text{ in } (\mathbf{m}_t, \mathbf{f}_t^n)\}$$

is defined. Then $t \vDash \alpha$ iff for all models $\mathbf{m}_t'$ in $\mathbf{f}_t^n(\mathbf{d}_t)$ we have $\mathbf{m}_t' \vDash \psi$.

References

[1] Abadi, M., 'Logic in Access Control', *Logic in Computer Science*, IEEE Computer Society, 2003, pp. 228–233.

[2] Abadi, M., 'Access Control in a Core Calculus of Dependency', *Electr. Notes Theor. Comput. Sci.*, 172: 5–31, 2007.

[3] Abadi, M., 'Variations in Access Control Logic', in R. van der Meyden and L. van der Torre, (eds.), *Deontic Logic in Computer Science*, vol. 5076 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 96–109.

[4] Abadi, M., M. Burrows, B. W. Lampson, and G. D. Plotkin, 'A Calculus for Access Control in Distributed Systems', in J. Feigenbaum, (ed.), *CRYPTO*, vol. 576 of *Lecture Notes in Computer Science*, Springer, 1991, pp. 1–23.

[5] Abadi, M., M. Burrows, B. W. Lampson, and G. D. Plotkin, 'A Calculus for Access Control in Distributed Systems', *ACM Trans. Program. Lang. Syst.*, 15 (4): 706–734, 1993.

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

[7] Barker, S., *The next 700 access control models or a unifying meta-model?*, SACMAT, 2009, pp. 187–196.

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

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

[10] Becker, M. Y., C. Fournet, and A. D. Gordon, 'Design and Semantics of a Decentralized Authorization Language', *CSF*, IEEE Computer Society, 2007, pp. 3–15.

[11] Bell, D. E., and L. J. LaPadula, 'Secure Computer System: Unified Exposition and Multics Interpretation', *MITRE-2997*, 1976.

[12] Bertino, E., C. Bettini, E. Ferrari, and P. Samarati, 'An Access Control Model Supporting Periodicity Constraints and Temporal Reasoning, *ACM Transactions on Database Systems*, 23 (3): 231–285, 1998.

[13] Bertolissi, C., M. Fernández, and S. Barker, 'Dynamic Event-Based Access Control as Term Rewriting', in S. Barker and G.-J. Ahn, (eds.), *DBSec*, vol. 4602 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 195–210.

[14] Genovese, V., D. M. Gabbay, G. Boella, and L. van der Torre, 'FSL – Fibred Security Language', *Normative Multi-Agent Systems*, number 09121 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009,

[15] Boella, G., D. M. Gabbay, V. Genovese, and L. van der Torre, 'Fibred Security Language', *Studia Logica* 92: 395–436, 2009.

[16] Clarke, D. E., 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.

[17] Ferraiolo, D. F., and V. Atluri, 'A meta model for access control: why is it needed and is it even possible to achieve?' *ACM Symposium on Access Control Models and Technologies - SACMAT*, 2008, pp. 153–154.

[18] Ferraiolo, D. F., 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.
[19] Gabbay, D. M., *Fibring Logics*, Oxford University Press, 1999.
[20] Halpern, J. Y., and V. Weissman, 'Using First-Order Logic to Reason about Policies', *ACM Trans. Inf. Syst. Secur.*, 11 (4), 2008.
[21] Harrison, M. A., W. L. Ruzzo, and J. D. Ullman, 'Protection in Operating Systems', *Commun. ACM*, 19 (8): 461–471, 1976.
[22] Jajodia, S., P. Samarati, M. Sapino, and V. Subrahmaninan, 'Flexible Support for Multiple Access Control Policies', *ACM TODS*, 26 (2): 214–260, 2001.
[23] Joshi, J., 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.
[24] Lampson, B. W., M. Abadi, M. Burrows, and E. Wobber, 'Authentication in Distributed Systems: Theory and Practice', *ACM Trans. Comput. Syst.*, 10 (4): 265–310, 1992.
[25] Landin, P. J., 'The Next 700 Programming Languages', *Commun. ACM*, 9 (3): 157–166, 1966.
[26] Li, N., B. N. Grosof, and J. Feigenbaum, 'Delegation logic: A logic-based approach to distributed authorization', *ACM Trans. Inf. Syst. Secur.*, 6 (1): 128–171, 2003.
[27] Li, N., J. C. Mitchell, and W. H. Winsborough, 'Design of a role-based trust-management framework', *IEEE Symposium on Security and Privacy*, 2002, pp. 114–130.
[28] Liau, C.-J., 'Belief, information acquisition, and trust in multi-agent systems–a modal logic formulation', *Artif. Intell.*, 149 (1): 31–60, 2003.
[29] OASIS, eXtensible Access Control Markup language (XACML), 2003. http://www.oasis-open.org/xacml/docs/.
[30] Sandhu, R. S., and Q. Munawer, 'How to Do Discretionary Access Control Using Roles', *ACM Workshop on Role-Based Access Control*, 1998, pp. 47–54.
[31] Weitzner, D. J., 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.

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

Guido Boella
Dept. of Computer Science
Università di Torino
C.so Svizzera, 185 - 10149 Torino, Italy
guido@di.unito.it

Dov M. Gabbay
Dept. of Computer Science
King's College London
The Strand, London, WC2A 2LS, UK
and
Dept. of Computer Science
Bar Ilan University
Ramat Gan, Israel
dov.gabbay@kcl.ac.uk

Valerio Genovese
Dept. of Computer Science
Università di Torino
C.So Svizzera, 185 - 10149 Torino, Italy
valerio.click@gmail.com