Rewrite Specifications of Access Control Policies in Distributed Environments

Clara Bertolissi[1] and Maribel Fernández[2]

[1]LIF, Université de Provence, Marseille, France
[2]King's College London, Dept. of Computer Science, London WC2R 2LS, U.K.
Clara.Bertolissi@lif.univ-mrs.fr, Maribel.Fernandez@kcl.ac.uk

Abstract. We define a *metamodel* for access control that takes into account the requirements of *distributed* environments, where resources and access control policies may be distributed across several sites. This distributed metamodel is an extension of the *category*-based metamodel proposed in previous work (from which standard centralised access control models such as MAC, DAC, RBAC, Bell-Lapadula, etc. can be derived). We use a *declarative formalism* in order to give an *operational semantics* to the distributed metamodel. We then show how various distributed access control models can be derived as instances of the distributed metamodel, including distributed models where each site implements a different kind of local access control model.
Key Words: Security Policies, Distributed Access Control, Operational Semantics, Rewriting.

1 Introduction

Using a formal specification for defining access control models and policies is particularly important in distributed contexts. The first attempts to develop formal theories to define and validate security policies (see, for instance, [16]) have used first-order theorem provers, purpose-built logics, or flow-analysis, but these approaches have limitations (as discussed for instance in [24]). More recently, rewriting techniques have been fruitfully exploited in the context of security protocols (see [6, 22, 2]), security policies controlling information leakage (see [21]), and access control policies (see [35, 12]). Along these lines, rewriting systems appear to be well adapted for providing a semantics for distributed access control mechanisms.

Over the last few years, a variety of access control models and languages for access control policy specification have been developed, often motivated by particular applications. We can mention the ANSI (hierarchical) role-based access control (H-RBAC) model [1], further extended with time and location constraints [17], the mandatory access control (MAC) model [9], the event-based access control (DEBAC) model [12], etc. A unifying metamodel for access control, which can be specialised for domain-specific applications, has been recently proposed in [4]. This unifying approach has advantages: for example, by identifying a core set of principles of access control, one can abstract away many of

the complexities that are found in specific access control models; this, in turn, helps to simplify the task of policy writing. A rewrite-based operational semantics for this metamodel was described in [14], where the expressive power of the metamodel is also demonstrated by showing that the above mentioned access control models can be derived as specific instances of the metamodel.

Based on the work reported in [14], in this paper we define a category-based access control metamodel for *distributed environments* where each component of the system preserves its autonomy, and provide a formal specification of access control evaluation using a declarative framework. The notion of distributed environment that we consider here is related to the notion of *federation* developed in the context of database systems (see for example [28, 19]), where a federated system integrates several databases while preserving their autonomy. In this paper we see a distributed system consisting of several sites, each with its own resources to protect, as a federation, and we focus on access control. More precisely, we define a framework for the specification (and enforcement) of global access control policies that take into account the local policies specified by each member of the federation.

A key aspect of our approach to access control, following [4], is to focus attention on the notion of a *category*. We regard categories as a primitive concept and we view classic types of groupings used in access control, like a role, a security clearance, a discrete measure of trust, etc., as particular instances of the more general notion of category. Here we will adapt the idea of categorisation to a distributed, federative setting. In a system with dispersed resources, classifications of entities may depend on the site to which the entity belongs. Moreover, permissions associated to categories of entities may also depend on the site where the category is defined. Therefore, we may want to use a distributed access control evaluation method, in addition to the central one proposed in [14], or we may want to combine the two.

In this paper, we first axiomatise a *distributed access control metamodel*, then give a rewrite-based operational semantics for this metamodel using the techniques introduced in [13], which allow us to deal in a uniform way with distributed systems where different access control policies are maintained locally. The rewrite-based specification that we describe enables access control policies to be defined in a declarative way and permits properties of access control policies to be proved.

Summarising, the main contributions of this paper are:

- the axiomatisation of a category-based access control metamodel that takes into account the requirements of distributed systems seen as federations in which each component preserves its autonomy;
- a declarative, rewrite-based operational semantics for the distributed access control metamodel (extending [14] to incorporate the notion of site and policies distributed across several sites);
- a formal operational semantics for access request evaluation, in centralised as well as in distributed contexts where information is shared;

– the generalisation of access request evaluation by integrating in the metamodel mechanisms for the resolution of policy conflicts.

The remainder of the paper is organised as follows. In Section 2, we recall some basic notions in term rewriting, and describe the centralised access control metamodel. In Section 3, we define the extension of the metamodel for distributed environments. In Section 4 we specify its operational semantics as a rewrite system. In Section 5, we show examples of combinations of access request answers issued by different sites using different local policies. We describe techniques for proving properties of access control policies in Section 6. In Section 7, we discuss related work. In Section 8, conclusions are drawn, and further work is suggested.

2 Preliminaries

2.1 Term Rewriting

A *signature* $\mathcal{F}$ is a finite set of *function symbols*, each with a fixed arity. $\mathcal{X}$ denotes a denumerable set of *variables* $X_1, X_2, \ldots$. The set $T(\mathcal{F}, \mathcal{X})$ of *terms* built up from $\mathcal{F}$ and $\mathcal{X}$ can be identified with a set of finite trees in the usual way. *Positions* are strings of positive integers denoting a path from the root to a node in the tree. The *subterm* of t at position p is denoted by $t|_p$ and the result of replacing $t|_p$ with u at position p in t is denoted by $t[u]_p$. This notation is also used to indicate that u is a subterm of t. $\mathcal{V}(t)$ denotes the set of variables occurring in t. A term is *linear* if variables in $\mathcal{V}(t)$ occur at most once in t. A term is *ground* if $\mathcal{V}(t) = \emptyset$. Substitutions are written as in $\{X_1 \mapsto t_1, \ldots, X_n \mapsto t_n\}$ where t_i is assumed to be different from the variable X_i. We use Greek letters for substitutions and postfix notation for their application.

Definition 1 (Rewrite step). *Given a signature $\mathcal{F}$, a* term rewrite system *on $\mathcal{F}$ is a set of rewrite rules $R = \{l_i \to r_i\}_{i \in I}$, where $l_i, r_i \in T(\mathcal{F}, \mathcal{X})$, $l_i \notin \mathcal{X}$, and $\mathcal{V}(r_i) \subseteq \mathcal{V}(l_i)$. A term t* rewrites *to a term u at position p with the rule $l \to r$ and the substitution σ, written $t \to_p^{l \to r} u$, or simply $t \to_R u$, if $t|_p = l\sigma$ and $u = t[r\sigma]_p$. Such a term t is called* reducible. *Irreducible terms are said to be in* normal form.

We denote by $\to_R^+$ (resp. $\to_R^*$) the transitive (resp. transitive and reflexive) closure of the rewrite relation $\to_R$. The subindex R will be omitted when it is clear from the context.

Example 1. Consider a signature for lists of natural numbers, with function symbols z (with arity 0) and s (with arity 1) to build numbers; nil (with arity 0), cons (with arity 2) and append (with arity 2) to build lists, $\in$ (with arity 2) to test the membership of a number in a list. The list containing the numbers 0 and 1 is written then as $\mathsf{cons}(\mathsf{z}, \mathsf{cons}(\mathsf{s}(\mathsf{z}), \mathsf{nil}))$, or simply $[\mathsf{z}, \mathsf{s}(\mathsf{z})]$ for short. We can specify list concatenation with the following rewrite rules: $\mathsf{append}(\mathsf{nil}, X) \to X$

and $\mathsf{append}(\mathsf{cons}(Y, X), Z) \rightarrow \mathsf{cons}(Y, \mathsf{append}(X, Z))$. Then we have a reduction sequence: $\mathsf{append}(\mathsf{cons}(\mathsf{z}, \mathsf{nil}), \mathsf{cons}(\mathsf{s}(\mathsf{z}), \mathsf{nil})) \rightarrow^* \mathsf{cons}(\mathsf{z}, \mathsf{cons}(\mathsf{s}(\mathsf{z}), \mathsf{nil}))$.

Boolean operators, such as disjunction, conjunction, and a conditional, can be specified using a signature that includes the constants true and false. The notation *if* b *then* s *else* t is syntactic sugar for the term $\mathsf{if\text{-}then\text{-}else}(b, s, t)$, with the rewrite rules: $\mathsf{if\text{-}then\text{-}else}(\mathsf{true}, X, Y) \rightarrow X$ and $\mathsf{if\text{-}then\text{-}else}(\mathsf{false}, X, Y) \rightarrow Y$.

For example, we can define the membership operator "$\in$" as follows: $\in(X, \mathsf{nil}) \rightarrow \mathsf{false}$, $\in(X, \mathsf{cons}(H, L)) \rightarrow$ *if* $X = H$ *then* true *else* $\in(X, L)$, where we assume "=" is a syntactic equality test defined by standard rewrite rules. We will often write $\in$ as an infix operator.

A term rewriting system R is *confluent* if for all terms t, u, v: $t \rightarrow^* u$ and $t \rightarrow^* v$ implies $u \rightarrow^* s$ and $v \rightarrow^* s$, for some s; it is *terminating* if all reduction sequences are finite. If all left-hand sides of rules in R are linear and rules are non-overlapping (i.e., there are no superpositions of left-hand sides) then R is orthogonal. Orthogonality is a sufficient condition for confluence [30].

For the approach to distributed access control that we propose later, we use distributed term rewrite systems (DTRSs) introduced in [12]. DTRSs are term rewrite systems where rules are partitioned into modules, each associated with a unique identifier, and function symbols are annotated with such identifiers. In a DTRS, we can associate a module to each site of a distributed system. For example, we may write f_ν to refer to the definition of the function symbol f in the site ν, where ν is a site identifier. We say that a rule $f(t_1, \ldots, t_n) \rightarrow r$ defines f; all the functions defined in a module are annotated with the same identifier. If a symbol is used in a rule without a site annotation, we assume the function is defined locally.

For more details on DTRSs, the reader can refer to [12].

2.2 Category-Based Metamodel

We briefly describe below the key concepts underlying the category-based metamodel of access control, henceforth denoted by $\mathcal{M}$. We refer the reader to [4] for a detailed description.

Informally, a category is any of several distinct classes or groups to which entities may be assigned. Entities are denoted uniquely by constants in a many sorted domain of discourse, including: a countable set $\mathcal{C}$ of categories, denoted c_0, c_1, ..., a countable set $\mathcal{P}$ of principals, denoted p_0, p_1, ..., a countable set $\mathcal{A}$ of named *actions*, denoted a_0, a_1, ..., a countable set $\mathcal{R}$ of *resource identifiers*, denoted r_0, r_1, ..., a finite set $\mathcal{A}uth$ of possible *answers* to access requests and a countable set $\mathcal{S}$ of *situational identifiers*. Situational identifiers are used to denote contextual or environmental information. We assume that principals that request access to resources are pre-authenticated. In the metamodel, the answer to a request may be one of a series of constants. For instance, the set $\mathcal{A}uth$ might include {grant, deny, $\mathsf{grant\text{-}if\text{-}obligation\text{-}is\text{-}satisfied}$, $\mathsf{undetermined}$}.

In addition to the different types of entities mentioned above, the metamodel includes the following relations:

– *Principal-category assignment:* $\mathcal{PCA} \subseteq \mathcal{P} \times \mathcal{C}$, such that $(p, c) \in \mathcal{PCA}$ iff a principal $p \in \mathcal{P}$ is assigned to the category $c \in \mathcal{C}$.
– *Permissions:* $\mathcal{ARCA} \subseteq \mathcal{A} \times \mathcal{R} \times \mathcal{C}$, such that $(a, r, c) \in \mathcal{ARCA}$ iff the action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$ can be performed by principals assigned to the category $c \in \mathcal{C}$.
– *Authorisations:* $\mathcal{PAR} \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{R}$, such that $(p, a, r) \in \mathcal{PAR}$ iff a principal $p \in \mathcal{P}$ can perform the action $a \in \mathcal{A}$ on the resource $r \in \mathcal{R}$.

Thus, $\mathcal{PAR}$ defines the set of authorisations that hold according to an access control policy that specifies $\mathcal{PCA}$ and $\mathcal{ARCA}$, as follows.

Definition 2 (Axioms). *The relation $\mathcal{PAR}$ satisfies the following core axiom, where we assume that there exists a relationship $\subseteq$ between categories; this can simply be equality, set inclusion (the set of principals assigned to $c \in \mathcal{C}$ is a subset of the set of principals assigned to $c' \in \mathcal{C}$), or an application specific relation may be used.*

$$(a1)\ \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},\ \forall c \in \mathcal{C},$$
$$(p, c) \in \mathcal{PCA} \wedge (\exists c' \in \mathcal{C}, c \subseteq c'\ \wedge (a, r, c') \in \mathcal{ARCA}) \Rightarrow (p, a, r) \in \mathcal{PAR}$$

The category-based metamodel of access control is based on the core axiom $(a1)$ in Def. 2. Operationally, this axiom can be realised through a set of function definitions, as shown in [14], where the information contained in the relations $\mathcal{PCA}$ and $\mathcal{ARCA}$ is modelled by functions pca and arca, respectively. The function pca returns the list of categories assigned to a principal, e.g. $\mathsf{pca}(\mathsf{p}) \rightarrow [\mathsf{c}]$, and arca returns a list of permissions assigned to a category, e.g. $\mathsf{arca}(\mathsf{c}) \rightarrow [(\mathsf{a_1}, \mathsf{r_1}), \ldots, (\mathsf{a_n}, \mathsf{r_n})]$.

Definition 3. *The rewrite-based specification of the axiom $(a1)$ in Def. 2 is given by the rewrite rule:*

$$(a2)\ \mathsf{par}(P, A, R) \rightarrow \mathit{if}\ (A, R) \in \mathsf{arca}^*(\mathsf{contain}(\mathsf{pca}(P)))\ \mathit{then}\ \mathsf{grant}\ \mathit{else}\ \mathsf{deny}$$

As the function name suggests, $\mathsf{contain}$ *computes the set of categories that contain any of the categories given in the list* $\mathsf{pca}(P)$*. For example, for a category* c*, this can be achieved by using a rewrite rule* $\mathsf{contain}([\mathsf{c}]) \rightarrow [\mathsf{c}, \mathsf{c_1}, \ldots, \mathsf{c_n}]$*. The function* $\in$ *is a membership operator on lists (see Section 2),* grant *and* deny *are answers, and* arca^* *generalises the function* arca *to take into account lists of categories:*

$$\mathsf{arca}^*(\mathsf{nil}) \rightarrow \mathsf{nil} \qquad \mathsf{arca}^*(\mathsf{cons}(C, L)) \rightarrow \mathsf{append}(\mathsf{arca}(C), \mathsf{arca}^*(L))$$

For optimisation purposes, one can compose the standard list concatenation operator append with a function removing the duplicate elements in the list.

An access request by a principal p to perform the action a on the resource r can then be evaluated simply by rewriting the term $\mathsf{par}(p, a, r)$ to normal form.

A range of access control models can be represented as specialised instances of the metamodel: see [14] for the specifications of traditional access control models, such as RBAC, DAC and MAC, (including the well-known Bell-LaPadula model), as well as the event-based model DEBAC.

The category-based metamodel defined in this section does not take into account distributed issues, such as conflicts that may arise between different local policies: all the relations defined are monolithic, and the axioms defining authorisations are valid in the whole system. In the next section we will extend and refine the metamodel in order to deal in a uniform way with access control policies for distributed systems.

3 A Distributed Category-Based Metamodel

We consider the same sets of entities as in the centralised metamodel $\mathcal{M}$. The set $\mathcal{S}$ of situational identifiers will now include identifiers for sites (or locations) which will be associated to resources or policies. For simplicity we will assume that $\mathcal{S}$ is just the set of locations that compose the distributed system. In other words, each $s \in \mathcal{S}$ identifies one of the components of the distributed system, seen as a federation. The set $\mathcal{P}$ includes the principals registered in any of the sites of the system. We assume that principals identities are globally known in the federation.

In addition to principal-category assignments, permissions, and authorisations (relations $\mathcal{PCA}$, $\mathcal{ARCA}$ and $\mathcal{PAR}$), we define a notion of forbidden operation (or banned action), modelled by the relation $\mathcal{BARCA}$, and a notion of non-authorised access, modelled by the relation $\mathcal{BAR}$:

- *Banned actions on resources:* $\mathcal{BARCA} \subseteq \mathcal{A} \times \mathcal{R} \times \mathcal{C}$, such that $(a, r, c) \in \mathcal{BARCA}$ iff the action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$ is forbidden for principals assigned to the category $c \in \mathcal{C}$.
- *Barred access:* $\mathcal{BAR} \subseteq P \times A \times \mathcal{R}$, such that $(p, a, r) \in \mathcal{BAR}$ iff performing the action $a \in \mathcal{A}$ on the resource $r \in \mathcal{R}$ is forbidden for the principal $p \in \mathcal{P}$.

Additionally, a relation $\mathcal{UNDET}$ could be defined if $\mathcal{PAR}$ and $\mathcal{BAR}$ are not complete, i.e., if there are access requests that are neither authorised nor denied (thus producing an undeterminate answer). These notions are not essential in centralised systems, but they are necessary in distributed systems for integrating partially specified policies, i.e. policies that may be "not applicable" to requests on resources that are out of their jurisdiction. Moreover, to take into account the fact that the system may be composed of several sites, with different policies in place at each site, we consider families of relations $\mathcal{PCA}_s$, $\mathcal{ARCA}_s$, $\mathcal{BARCA}_s$, $\mathcal{BAR}_s, \mathcal{UNDET}_s$ and $\mathcal{PAR}_s$ indexed by site identifiers. Intuitively, $\mathcal{PAR}_s$ (resp. $\mathcal{BAR}_s$) denotes the authorisations (resp. prohibitions) that are valid in the site s. The relation $\mathcal{PAR}$ defining the global authorisation policy will be obtained by composing the local policies defined by the relations $\mathcal{PAR}_s$ and $\mathcal{BAR}_s$ as indicated below. For instance, $\mathcal{PAR}$ could be defined as a union, but more sophisticated combinations are possible, in particular if policies in different sites may contain conflicting information.

The axioms for the distributed metamodel are given below; they can be seen as an extension of the axioms that define $\mathcal{M}$ (see Definition 2).

Definition 4 (Distributed Axioms). *In a distributed environment, the category based metamodel is defined by the following core axioms where we assume that there exists a relationship $\subseteq$ between categories; this can simply be equality, set inclusion (i.e., the set of principals assigned to $c \in \mathcal{C}$ is a subset of the set of principals assigned to $c' \in \mathcal{C}$), or an application specific relation may be used.*

$$(b1)\ \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},\ \forall c \in \mathcal{C},\ \forall s \in \mathcal{S}$$
$$(p, c) \in \mathcal{PCA}_s \wedge (\exists c' \in \mathcal{C}, c \subseteq c'\ \wedge (a, r, c') \in \mathcal{ARCA}_s) \Rightarrow (p, a, r) \in \mathcal{PAR}_s$$

If the relation $\mathcal{BARCA}$ is admitted in a site s, then the following axioms should be included:

$$(c1)\ \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},\ \forall c \in \mathcal{C},\ \forall s \in \mathcal{S}$$
$$(p, c) \in \mathcal{PCA}_s \wedge (\exists c' \in \mathcal{C}, c \subseteq c'\ \wedge (a, r, c') \in \mathcal{BARCA}_s) \Rightarrow (p, a, r) \in \mathcal{BAR}_s$$

$$(d1)\ \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},\ \forall c \in \mathcal{C},\ \forall s \in \mathcal{S}$$
$$(p, c) \in \mathcal{PCA}_s\ \wedge\ (a, r, c) \notin \mathcal{ARCA}_s\ \wedge\ (a, r, c) \notin \mathcal{BARCA}_s \Rightarrow (p, a, r) \in \mathcal{UNDET}_s$$
$$(e1)\ \forall s \in \mathcal{S},\quad \mathcal{ARCA}_s \cap \mathcal{BARCA}_s = \emptyset$$

Finally, the axioms below describe the global authorisation relation in terms of the local policies defined at each site:

$$(f1)\ \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},$$
$$(p, a, r) \in \mathcal{OP}_{par}(\{\mathcal{PAR}_s, \mathcal{BAR}_s \mid s \in S\}) \Rightarrow (p, a, r) \in \mathcal{PAR}$$
$$(g1)\ \forall p \in \mathcal{P},\ \forall a \in \mathcal{A},\ \forall r \in \mathcal{R},$$
$$(p, a, r) \in \mathcal{OP}_{bar}(\{\mathcal{PAR}_s, \mathcal{BAR}_s \mid s \in S\}) \Rightarrow (p, a, r) \in \mathcal{BAR}$$
$$(h1)\ \mathcal{PAR} \cap \mathcal{BAR} = \emptyset$$

According to these axioms, the result of an access request may be different depending on the site where the request is evaluated (since each site has its own authorisation policy defined by the local relation $\mathcal{PAR}_s$). The relation $\mathcal{UNDET}_s \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{R}$ is such that $(p, a, r) \in \mathcal{UNDET}_s$ iff the action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$ is neither allowed nor forbidden for the principal $p \in \mathcal{P}$ at site $s \in \mathcal{S}$. The final authorisation is computed specialising the definition of the operators $\mathcal{OP}_{par}$ and $\mathcal{OP}_{bar}$, according to the application requirements. For instance, consider a system with two sites $s, t \in \mathcal{S}$, we may have $\mathcal{OP}_{bar} = (\mathcal{BAR}_s \vee \mathcal{BAR}_t)$ and $\mathcal{OP}_{par} = ((\mathcal{PAR}_s / \mathcal{BAR}_t) \vee (\mathcal{PAR}_t / \mathcal{BAR}_s))$. This corresponds to a union operator giving priority to deny, according to the "deny takes precedence principle" [26] (i.e. access is denied if it is denied by any of the component policies).

Also, if the information available in a site s is not sufficient to grant a principal the right to access a certain resource, but does not ban the access either (i.e., the outcome of the request evaluation cannot be determined at s), then the access request may need to be processed in another site of the system. We can model this kind of policy definition with the axioms above using the operator $\mathcal{OP}_{par} = (\mathcal{PAR}_s \vee (\mathcal{PAR}_t / \mathcal{PAR}_s)\}$ and $\mathcal{OP}_{bar} = (\mathcal{BAR}_s \vee (\mathcal{BAR}_t / \mathcal{BAR}_s))$. This corresponds to the precedence operator, as defined e.g. in [15].

The distributed metamodel refines and extends $\mathcal{M}$ as defined in Section 2.2 because different access control policies (possibly using *different access control models*) may be defined at different sites, and access requests can be evaluated using any combination of local access control policy as well as a global, centrally defined policy (for instance, if the system has a central policy at site ς, then $\mathcal{OP}_{par} = (\mathcal{PAR}_{\varsigma})$ and $\mathcal{OP}_{bar} = (\mathcal{BAR}_{\varsigma})$).

Intuitively, the axioms preclude inconsistent specifications (axioms $e1$ and $h1$) and define authorised actions as follows:

> *A principal $p \in \mathcal{P}$ is permitted to perform an action $a \in \mathcal{A}$ on a resource $r \in \mathcal{R}$ if*
> i) *at the site where the request is evaluated, p is assigned to a category $c \in \mathcal{C}$ to which a access on r has been locally assigned (axiom b_1), or*
> ii) *in the global policy, p is assigned to a category $c \in \mathcal{C}$ to which a access on r has been granted (axiom f_1).*

4 Operational Semantics

The operational semantics of the distributed model will be defined by extending the functions presented in Definition 3 and in addition a new function barca, corresponding to the relation $\mathcal{BARCA}$ in Section 3, will be introduced to represent explicitly forbidden actions on resources. We recall that functions with no site annotations are assumed to be defined locally. The function barca_s returns the list of prohibitions assigned to a category in a site s:
$\mathsf{barca}_s(\mathsf{c}) \rightarrow [(\mathsf{a}_1, \mathsf{r}_1), \ldots, (\mathsf{a}_n, \mathsf{r}_n)]$.

If for a given category c, a pair (a, r) is neither in $\mathsf{arca}_s(c)$ nor in $\mathsf{barca}_s(c)$, then we say that such access privilege is undefined and consequently the corresponding access request is *undeterminate*. This permits a finer-grained evaluation of access requests: the constant undet is a possible answer, at the same level as grant and deny. We assume that, in a correct specification, for the same category c, $\mathsf{arca}_s(c)$ and $\mathsf{barca}_s(c)$ do not have elements in common, i.e., if $(a_i, r_j) \in \mathsf{arca}_s(c)$, then $(a_i, r_j) \notin \mathsf{barca}_s(c)$, and viceversa, for any $i, j \in \mathbb{N}$, as required by axiom $(e1)$. The rewrite-based specification of the extended axioms of Section 3 follows.

Definition 5. *In a distributed environment, the rewrite-based specification of the axiom $(b1)$ in Def. 4 is given by the rewrite rule:*

$$(b2)\ \mathsf{par}_s(P, A, R) \rightarrow if\ (A, R) \in \mathsf{arca}^*_s(\mathsf{contain}(\mathsf{pca}_s(P)))\ then\ \mathsf{grant}\ else\ \mathsf{deny}$$

where the function $\in$ is a membership operator on lists (see Section 2), grant *and* deny *are answers, and arca^*_s is the function defining the assignment of privileges to categories, as in the previous section.*

If we consider the extended axioms $(c1)$, $(d1)$ in Def. 4, which involve a $\mathcal{BARCA}$ relation, the function par *is defined as:*

$$(c2, d2)\ \mathsf{par}_s(P, A, R) \rightarrow if\ (a, r) \in \mathsf{arca}^*_s(\mathsf{contain}(\mathsf{pca}_s(p)))then\ \mathsf{grant}\ else\ if\ (a, r) \in \mathsf{barca}^*_s(\mathsf{contain}(\mathsf{pca}_s(p)))then\ \mathsf{deny}\ else\ \mathsf{undet}$$

where barca_s^* *generalises the previously mentioned function* barca_s *to take into account lists of categories instead of a single category:*

$$\mathsf{barca}^*(\mathsf{nil}) \rightarrow \mathsf{nil} \qquad \mathsf{barca}^*(\mathsf{cons}(C, L)) \rightarrow \mathsf{append}(\mathsf{barca}(C), \mathsf{barca}^*(L))$$

As already mentioned, the functions arca_s *and* barca_s *should satisfy axiom* $(e1)$*, but note that even if axiom* $(e1)$ *is not satisfied, the operational semantics defined for* par *above is consistent: it gives priority to positive authorisations.*

The axioms $(f1)$ *and* $(g1)$ *are realised by the following rewrite rule (implementing* $\mathcal{OP}_{par}$*):*

$$(f2, g2)\ \mathsf{authorised}(P, A, R, s_1, \ldots, s_n) \rightarrow \mathsf{fauth}(op, \mathsf{par}_{s_1}(p, a, r), \ldots, \mathsf{par}_{s_n}(p, a, r))$$

where the function fauth *combines the results into a final access authorisation according to the operator op.*

The axioms $(f1)$ and $(g1)$ can be implemented in several ways. The version chosen in the definition above corresponds to a very general rewrite rule that can be used for evaluating an access request in a single central site, if $n = 1$ and the operator *op* is the identity, as well as for evaluating combinations of answers (with a suitable operator *op*) from n different local policies. An alternative can be to specify an authorised rewrite rule for any specific combination operator. For example, if we consider two sites s_1 and s_2, for the precedence operator previously mentioned, we may have

$$\begin{aligned}\mathsf{authorised}(P, A, R, S_1, S_2) \rightarrow\ & \mathit{if}\ \mathsf{par}_{s_1}(P, A, R) = \mathsf{grant}\ \mathit{or}\ \mathsf{par}_{s_1}(P, A, R) = \mathsf{deny} \\ & \mathit{then}\ \mathsf{par}_{s_1}(P, A, R)\ \mathit{else}\ \mathsf{par}_{s_2}(P, A, R)\end{aligned}$$

In this case priority is given to local evaluation in site s_1, external evaluation in site s_2 being executed only when the local policy in s_1 is not able to give as answer grant or deny. However, when dealing with policy combinations, it is unlikely to find a unique evaluation strategy that works for every possible scenario. A suitable policy integration mechanism depends on the requirements of the application and the involved parties. Distributed evaluation of access requests and combination of authorisation answers is further discussed in the next section. An overview of the rules modelling the distributed version of $\mathcal{M}$ is given in Table 1.

4.1 Evaluating access requests

The novelty of the distributed metamodel lies in the fact that different local policies can be defined and combined in this framework in a smooth and uniform manner.

Evaluation initially takes place in the site where the request is issued, using the local function authorised (see rule *Aut* in Table 1). The local rule *Aut* specifies a number of sites s_i, $i = 1 \ldots n$, where the request may be passed and evaluated by the corresponding function par_{s_i}. For defining the sites of the evaluation, we

may use functions like e.g. $\mathsf{psite}(p)$, which returns the site where the principal p is registered, or $\mathsf{rsite}(r)$ which returns the site where the resource r is located. In this way, access requests can be evaluated in a predefined central site, or priority can be given to local evaluation, or more elaborated combinations of access answers can be implemented.

In order to specify a particular policy at a site s, e.g. RBAC, MAC or DEBAC, it is sufficient to specialise locally the functions arca_s, barca_s, pca_s, $\mathsf{contain}_s$. Their definition in Table 1 can be adapted to the application we want to consider. Thus, for example, if we want to express a hierarchical RBAC policy at site s, the function arca_s will return the permissions associated to each defined role and $\mathsf{contain}_s$ will return the roles senior to a given role, according to the hierarchy specified in the model (see [14] for more application examples).

Thus, in the distributed metamodel, the request can be passed to other sites (with possibly different local policies) and evaluated in a distributed way. The generic rule (Aut), implementing the axioms of Section 3, will then integrate the different local access request answers to provide a final authorisation decision. More precisely, the distributed answers are collected and combined by using the specific rule $(Fauth)$ which is locally defined for an operator op. For example, consider a principal in an international organisation belonging to the U.K division and asking for access to the French division. In this case, we want to evaluate the access authorisation in the U.K. site, where the principal is registered, and also in the French site, the two sites possibly having different policies. Access will be permitted if the policies in both sites return a grant answer (denied if at least one policy denies the access). In our metamodel, we can combine evaluations from different sources in a flexible way by refining the definition of the fauth function (rule *Fauth*). In our example, we would use a fauth function with a union operator:

$$\begin{array}{ll} \mathsf{fauth}(\mathsf{union},\mathsf{deny},\mathsf{X}) \rightarrow \mathsf{deny} & \mathsf{fauth}(\mathsf{union},\mathsf{X},\mathsf{deny}) \rightarrow \mathsf{deny} \\ \mathsf{fauth}(\mathsf{union},\mathsf{undet},\mathsf{undet}) \rightarrow \mathsf{undet} & \mathsf{fauth}(\mathsf{union},\mathsf{grant},\mathsf{grant}) \rightarrow \mathsf{grant} \\ \mathsf{fauth}(\mathsf{union},\mathsf{undet},\mathsf{grant}) \rightarrow \mathsf{undet} & \mathsf{fauth}(\mathsf{union},\mathsf{grant},\mathsf{undet}) \rightarrow \mathsf{undet} \end{array}$$

Following the same idea, the rewrite system can be adapted to a variety of algebraic commutative and associative operators, as well as operators in the style of those defined in [15] (Subtraction, Precedence, etc.). For more elaborated operators, higher-order features can be added to the model, along the lines of [10].

5 Examples

We give next more detailed examples to demonstrate the expressive power of the distributed metamodel.

Example 2. (Combining RBAC and DEBAC policies) Consider an organisation composed of several commercial branches, all depending from a central department. Assume a principal p is the director of the London branch and has currently been assigned the management of a project Proj involving several U.K. branches. The information on the project is stored at the departmental level.

Table 1. Rewrite specification of the metamodel

Par_s	$\mathsf{par}(P, A, R)$	$\rightarrow$ *if* $(A, R) \in \mathsf{arca}^*(\mathsf{contain}(\mathsf{pca}(P)))$ *then* grant *else* *if* $(A, R) \in \mathsf{barca}^*(\mathsf{contain}(\mathsf{pca}(P)))$ *then* deny *else* undet
$Arca^*$	$\mathsf{arca}^*(\mathsf{cons}(C, L))$	$\rightarrow \mathsf{append}(\mathsf{arca}(C), \mathsf{arca}^*(L))$
$Arca^*$	$\mathsf{arca}^*(\mathsf{cons}(C, \mathsf{nil}))$	$\rightarrow \mathsf{nil}$
$Barca^*$	$\mathsf{barca}^*(\mathsf{cons}(C, L))$	$\rightarrow \mathsf{append}(\mathsf{barca}(C), \mathsf{barca}^*(L))$
$Barca^*$	$\mathsf{barca}^*(\mathsf{cons}(C, \mathsf{nil}))$	$\rightarrow \mathsf{nil}$
Pca	$\mathsf{pca}(\mathsf{p})$	$\rightarrow [\mathsf{c}]$
$Arca$	$\mathsf{arca}(\mathsf{c})$	$\rightarrow [(\mathsf{a}_1, \mathsf{r}_1), \ldots, (\mathsf{a}_k, \mathsf{r}_k)]$
$Barca$	$\mathsf{barca}(\mathsf{c})$	$\rightarrow [(\mathsf{a}_l, \mathsf{r}_l), \ldots, (\mathsf{a}_t, \mathsf{r}_t)]$
$Contain$	$\mathsf{contain}(\mathsf{c})$	$\rightarrow [\mathsf{c}, \mathsf{c}_1, \ldots, \mathsf{c}_n]$

Aut	$\mathsf{authorised}(P, A, R, s_1, \ldots, s_n)$	$\rightarrow \mathsf{fauth}(\mathsf{op}, \mathsf{par}_{s_1}(P, A, R), \ldots, \mathsf{par}_{s_n}(P, A, R))$
$Fauth$	$\mathsf{fauth}(op, \mathsf{par}_{s_1}(p, a, r), \ldots, \mathsf{par}_{s_n}(p, a, r))$	$\rightarrow \mathsf{answ}$ *with* $\mathsf{answ} \in \mathcal{A}\mathsf{uth}$

Assume a principal wants to read the balance sheet of the project, and, according to the policy of the organisation, only the leader of the project has the right to access it. In our model, this corresponds to the presence of the pair $(\mathsf{read}, \mathsf{balanceProj})$ in the privileges associated to the category "leader of the project", $\mathsf{arca}_\delta(\mathsf{leaderProj}) \rightarrow [(\mathsf{read}, \mathsf{balanceProj}), \ldots]$, stored at the departmental site δ.

In the distributed evaluation model, the request from p to read the balance of Proj is first treated by the site ν where the request is issued, which passes it to the local branch where p is registered, say π, and to departmental site δ.

$$\mathsf{authorised}_\nu(\mathsf{p}, \mathsf{read}, \mathsf{balanceProj}, \mathsf{psite}(\mathsf{p}), \mathsf{dept}(\mathsf{p}))) \rightarrow \mathsf{authorised}_\nu(\mathsf{p}, \mathsf{read}, \mathsf{balanceProj}, \pi, \delta)$$

Suppose the organisation implements locally for each branch a role-based policy and at the departmental level an event-based policy, to deal with dynamic changes. We refer to [14] for the definition of individual RBAC and DEBAC policies in the metamodel. Here we consider the distributed model where the function fauth combines policies according to the precedence operator (see Section 4), focusing on the evaluation of the access request locally. Assume we have

$$\mathsf{pca}_\pi(\mathsf{p}) \rightarrow [\mathsf{director}]$$
$$\mathsf{arca}_\pi(\mathsf{director}) \rightarrow [(\mathsf{read}, \mathsf{report}), (\mathsf{write}, \mathsf{report}), \ldots,]$$
$$\mathsf{barca}_\pi(\mathsf{director}) \rightarrow [(\mathsf{delete}, \mathsf{trail}), \ldots,]$$

No privileges associated to the project Proj are present in the policy defined for branch π. Therefore, the local request evaluation will produce as result undet:

local information is not sufficient to produce an authorisation or a denial of access, $\mathsf{par}_\pi(\mathsf{p},\mathsf{read},\mathsf{balanceProj}) \rightarrow^* \mathsf{undet}$.

The request is then evaluated at the departmental site by the function par_δ which calls again the function pca to compute the category associated to principal p. The difference with respect to the local par_π function is that the policy associated to the central site is a DEBAC policy, and thus the resulting category associated to p may be different. The DEBAC policy defined in δ computes the category of a principal according to a history of events h_δ. In such history, among others, we have recorded the events relevant to the project Proj, such as the nomination of the managers and participants to the project. Thus p's category in site ς may be computed using rules:

$$
\begin{array}{ll}
\mathsf{pca}(P) & \rightarrow \mathsf{categ}(P, \mathsf{h}) \\
\mathsf{categ}(P, \mathsf{nil}) & \rightarrow \mathsf{Participant} \\
\mathsf{categ}(P, \mathsf{cons}(\mathsf{event}(E, P, \mathsf{inchargeProj}, T), H)) & \rightarrow if\ P \in \mathsf{Managers}(\mathsf{Proj}) \\
 & \quad then\ [\mathsf{LeaderProj}]) \\
 & \quad else\ [\mathsf{ParticipantProj}]
\end{array}
$$

meaning that the events that happened at some previous time t involving p in the project Proj determine the category of p. $\mathsf{Managers}(P)$ returns the list of managers associated to project P that we suppose registered locally in site δ.

So, according to the DEBAC policy, we have $\mathsf{pca}_\varsigma(\mathsf{p}) \rightarrow [\mathsf{LeaderProj}]$ because in the history of events there is a tuple indicating that p became a leader of the project, and since the pair (read, balanceProj) is in the privileges of the category LeaderProj, we have $\mathsf{par}_\varsigma(\mathsf{p},\mathsf{read},\mathsf{balanceProj}) \rightarrow^* \mathsf{grant}$ and thus finally the reduction $\mathsf{authorised}_\nu(\mathsf{p}, \mathsf{read}, \mathsf{balanceProj}, \pi, \delta) \rightarrow^* \mathsf{grant}$.

Example 3. (Combining RBAC and Bell-Lapadula policies)

Consider a principal working in an organisation where employees share an electronic agenda a, maintained on a server ν. Suppose this organisation adopts an RBAC policy for the employees, and moreover uses a specific Bell-Lapadula policy on the agenda in site ν. Consider the request of editing the agenda by the principal p. In this case, p has to be an employee of the organisation to access the agenda and moreover p must respect the "no read up" and "write only at the subject level" rules of the Bell-Lapadula policy in order to edit it. In the metamodel, we will use a union operator giving priority to deny (see Section 4.1) to meet the requirements of this distributed scenario. At the local level, we have an RBAC policy implemented in the site where the principal is registered, say π. The policy provides rules such as

$$
\begin{array}{l}
\mathsf{pca}_\pi(\mathsf{p}) \rightarrow [\mathsf{employee}] \\
\mathsf{arca}_\pi(\mathsf{employee}) \rightarrow [(\mathsf{read}, \mathsf{report}), (\mathsf{write}, \mathsf{a_{all}}), (\mathsf{read}, \mathsf{a_{all}}] \\
\mathsf{barca}_\pi(\mathsf{employee}) \rightarrow [(\mathsf{write}, \mathsf{report}), \ldots,]
\end{array}
$$

where *all* stands for any section of the agenda. We have in addition a Bell-Lapadula policy local to site ν. Assume we have three different levels (top-secret, secret and public) and three different sections of the agenda ts, s and p that can

be modified according to the category associated to the principal. The privileges depend thus on the secrecy level:

$$\begin{array}{l}\mathsf{arca}_\nu(\mathsf{top\_secret}) \rightarrow [(\mathsf{read}, \mathsf{a_{ts}}), (\mathsf{write}, \mathsf{a_{ts}}), (\mathsf{read}, \mathsf{a_s}), (\mathsf{read}, \mathsf{a_p}),]\\ \mathsf{barca}_\nu(\mathsf{top\_secret}) \rightarrow [(\mathsf{write}, \mathsf{a_s}), (\mathsf{write}, \mathsf{a_p})]\\ \ldots\\ \mathsf{arca}_\nu(\mathsf{public}) \rightarrow [(\mathsf{write}, \mathsf{a_p}), (\mathsf{read}, \mathsf{a_p}),]\\ \mathsf{barca}_\nu(\mathsf{public}) \rightarrow [(\mathsf{write}, \mathsf{a_s}), (\mathsf{write}, \mathsf{a_{ts}}), (\mathsf{read}, \mathsf{a_s}), (\mathsf{read}, \mathsf{a_{ts}})]\end{array}$$

Assume p is assigned to the public level, $\mathsf{pca}_\nu(\mathsf{p}) \rightarrow [\mathsf{public}]$, and asks for editing a section $\mathsf{a_s}$ in the agenda.

The request evaluation starts by calling the authorised function local to the site where the request is issued

$$\begin{array}{l}\mathsf{authorised}(\mathsf{p}, \mathsf{write}, \mathsf{a_s}, \mathsf{psite}(\mathsf{p}), \mathsf{rsite}(\mathsf{a}))\\ \rightarrow^* \mathsf{fauth}(\mathsf{union}, \mathsf{par}_\pi(\mathsf{p}, \mathsf{write}, \mathsf{a_s}), \mathsf{par}_\nu(\mathsf{p}, \mathsf{write}, \mathsf{a_s}))\end{array}$$

We consider first the evaluation of the request in the site $\mathsf{psite}(\mathsf{p}) \rightarrow \pi$. In this case, we have $\mathsf{par}_\pi(\mathsf{p}, \mathsf{write}, \mathsf{a_s}) \rightarrow^* \mathsf{grant}$ since p is indeed an employee of the organisation and as such has access to the shared agenda. We consider now the finer-grained evaluation of the request which is performed locally to site ν. We have $\mathsf{par}_\nu(\mathsf{p}, \mathsf{write}, \mathsf{a_s}) \rightarrow^* \mathsf{deny}$ since the pair $(\mathsf{write}, \mathsf{a_s})$ is in the list of prohibitions of the category public to which the principal belongs. Thus finally the access to principal p at the secret section of the agenda will be denied

$$\mathsf{authorised}(\mathsf{p}, \mathsf{write}, \mathsf{a_s}, \mathsf{psite}(\mathsf{p}), \mathsf{rsite}(\mathsf{a})) \rightarrow^* \mathsf{fauth}(\mathsf{union}, \mathsf{grant}, \mathsf{deny}) \rightarrow^* \mathsf{deny}$$

6 Policy Analysis: Proving Properties of Policies

Specifying access control policies via term rewriting systems, which have a formal semantics, has the advantage that this representation admits the possibility of proving properties of policies, and this is essential for policy acceptability [34]. Rewriting properties may be used to demonstrate satisfaction of essential properties of policies, such as:

Totality: Each access request from a valid principal p to perform a valid action a on a resource r receives an answer (e.g., grant, deny, undeterminate).

Consistency: For any $p \in \mathcal{P}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$, at most one result is possible for an authorisation request $\mathsf{par}(p, a, r)$.

Soundness and Completeness: For any $p \in \mathcal{P}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$, an access request by p to perform the action a on r is granted if and only if p belongs to a category that has the permission (a, r).

Totality and consistency can be proved, for policies defined as term rewriting systems, by checking that the rewrite relation generated by the rules used in a specific instance of the metamodel is confluent and terminating. Termination ensures that all access requests produce a result (e.g. a result that is not grant or deny is interpreted as undet) and confluence ensures that this result is unique.

The soundness and completeness of a policy can be checked by analysing the normal forms of access requests.

Confluence and termination of rewriting are undecidable properties in general, but there are several results available that provide sufficient conditions for these properties to hold. For instance, a *hierarchical* term rewriting system is terminating if the basis of the hierarchy is terminating and non-duplicating (i.e., rules do not duplicate variables in the right-hand side) and in the next levels of the hierarchy the recursive functions are defined by rules that satisfy a general scheme of recursion, where recursive calls on the right-hand sides of rules are made on subterms of the left-hand side and there are no mutually recursive functions [23]. Using the metamodel, the full definition of the policy can be seen as a hierarchical rewrite system, where the basis includes the set of constants identifying the main entities in the model (e.g., principals, categories, etc.) as well as the set of auxiliary data structures (such as booleans, lists) and functions on these data structures. The next level in the hierarchy contains the parameter functions of the model, namely pca, arca, barca, arca*, barca*, contain. Finally the last level of the hierarchy consists of the definition of the function par, and the functions authorised and fauth.

For instance, for the examples in this paper, provided the auxiliary functions in the basis of the hierarchy are defined by non-duplicating and terminating rewrite rules, the system is terminating (notice that the functions par, authorised and fauth are not recursive).

Confluence can be proved in various ways. Orthogonal systems are confluent, as shown by Klop [30]. A less restrictive condition, for systems that terminate, is the absence of critical pairs (the latter, combined with termination implies confluence by Newman's lemma [33]).

7 Related Work

There are several works in the literature using term rewriting to model access control problems. Koch et al. [31] use graph transformation rules to formalise RBAC, and more recently, [5, 35, 12] use term rewrite rules to model particular access control models and to express access control policies. Our work addresses similar issues to [5, 31, 35, 12], but is based on a notion of a meta-model of access control for distributed environments, from which various models can be derived, instead of formalising a specific kind of model such as RBAC or DEBAC.

Several proposals for general models and languages for access control have already been described in the literature, but neither of them provides the level of generality of the category-based approach. For example, the Generalised TRBAC model [29] and ASL [25] aim at providing a general framework for the definition of policies, however they focus essentially on the notion of users, groups and roles (interpreted as being synonymous with the notion of job function). Li et al.'s *RT* family of role-trust models [32] provides a general framework which can be specialised for defining specific policy requirements (in terms of credentials). The

category-based metamodel, however, can be instantiated to include concepts like times, events, actions and histories that are not included as elements of RT.

The metamodel defined in [4] incorporates the notion of site like our distributed metamodel, but there is no explicit definition of "forbidden" action and the issue of combinations of different policies is not addressed.

The metamodel that we have described is more expressive than any of the Datalog-based languages that have been proposed for distributed access control (see [3, 27, 20, 7]); these languages, being based on a monotonic semantics, are not especially well suited for representing dynamically changing distributed access request policies.

Another work dealing with decentralised systems is reported in [8], where the authors propose the constraint logic programming language SecPal for specifying a wide range of authorisation policies and credentials, using predicates defined by clauses. Inspired by this work, the DKAL authorisation language was proposed, based on existential fixed-point logic. In our approach, we focus on the definition of a general metamodel suitable for distributed systems rather than on the design of a specification language, but we give an operational semantics for the metamodel which can be instantiated further to derive specific access control models and policies.

The notion of Federated Systems can also be related to our work. In [28, 19] local systems evolving independently cooperate in a distributed architecture called federation. In our model, we can simulate this behaviour specifying local sets of registered users and defining rules for treating requests from external users at the federation level (associated to a remote site).

8 Conclusions and Further Work

We have given a rewrite-based specification of a distributed metamodel of access control that is based on general, common concepts of access control models. The term rewriting approach can be used to give a meaningful formal semantics to policies in the case of both centralised and distributed computer systems. In addition, operators for defining combinations of policies can be smoothly integrated in our framework. In future work, we plan to develop the algebra of operators used for integrating policies, along the lines of [10, 15], in order to express arbitrary combinations of policies at a finer granularity.

Rewrite rules provide a declarative specification of access control requirements which facilitates the task of proving properties of policies. Also, term rewriting rules provide a well-defined, executable specification of the access control policy. We plan to investigate the design of languages for policy specification and the practical implementation of category-based policies, defined as instances of our metamodel; in particular, an access control policy may be transformed into a MAUDE [18] program by adding type declarations for the function symbols and variables used and by making minor syntactical changes (see [11]). The distributed features, like the notion of site and local evaluation, can be reproduced in different Maude modules in which rewrite rules are partitioned.

References

1. ANSI. RBAC, 2004. INCITS 359-2004.
2. A. Armando et al. The Avispa Tool for the automated validation of internet security protocols and applications. *In Proc. of CAV 2005*, Computer Aided Verification, LNCS 3576, Springer Verlag.
3. J. Bacon, K. Moody, and W. Yao. A model of OASIS RBAC and its support for active security. *TISSEC*, 5(4):492–540, 2002.
4. S. Barker. The next 700 access control models or a unifying meta-model? In *Proc. of ACM Int. Conf. SACMAT 2009*, pages 187–196. ACM Press, 2009.
5. S. Barker and M. Fernández. Term rewriting for access control. In *Data and Applications Security. Proc. of DBSec'2006*, LNCS. Springer-Verlag, 2006.
6. G. Barthe, G. Dufay, M. Huisman, and S. Melo de Sousa. Jakarta: a toolset to reason about the JavaCard platform. In *Proc. of e-SMART'01*, volume 2140 of *LNCS*. Springer-Verlag, 2002.
7. M. Becker and P. Sewell. Cassandra: Distributed access control policies with tunable expressiveness. In Proc. of *POLICY 2004*, pages 159–168, 2004.
8. M. Y. Becker, C. Fournet, and A. D. Gordon. SecPAL: Design and Semantics of a Decentralized Authorization Language. *Journal of Computer Security* 18(4), 597–643, IOS Press, 2010.
9. D. E. Bell and L. J. LaPadula. Secure computer system: Unified exposition and multics interpretation. *MITRE-2997*, 1976.
10. C. Bertolissi and M. Fernández. A rewriting framework for the composition of access control policies. In *Proc. of PPDP 2008, Valencia, 2008.* ACM Press, 2008.
11. C. Bertolissi and M. Fernández. Time and location based services with access control. In *Proc. of NMTS'2008.* IEEEXplore, 2008.
12. C. Bertolissi, M. Fernández, and S. Barker. Dynamic event-based access control as term rewriting. In *Data and Applications Security. Proc. of DBSec'2007*, volume 4602 of *LNCS*. Springer-Verlag, 2007.
13. C. Bertolissi and M. Fernández. Distributed event-based access control. *International Journal of Information and Computer Security, Special Issue: selected papers from Crisis 2008*, volume 3, 2009.
14. C. Bertolissi and M. Fernández. Category-based authorisation models: operational semantics and expressive power. In *Proc. of ESSOS 2010*, LNCS. Springer, 2010.
15. P. Bonatti, S. de Capitani di Vimercati, and P. Samarati. A modular approach to composing access control policies. In *Proc. of CCS '00*, pages 164–173. ACM Press, 2000.
16. P. A. Bonatti and P. Samarati. Logics for authorization and security. In *Logics for Emerging Applications of Databases*, pages 277–323. Springer, 2003.
17. S. M. Chandran and J. B. D. Joshi. Lot-rbac: A location and time-based rbac model. In *Proc. of WISE 2005*, *LNCS* vol. 3806, pages 361–375. Springer, 2005.
18. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. The Maude 2.0 system. In *Proc. of RTA 2003*, LNCS volume 2706, pages 76–87. Springer-Verlag, 2003.
19. S. De Capitani di Vimercati, and P. Samarati. Authorization Specification and Enforcement in Federated Database Systems. In Journal of Computer Security, vol. 5, pages 155–188, 1997.
20. J. DeTreville. Binder, a logic-based security language. In *Proc. IEEE Symposium on Security and Privacy*, pages 105–113, 2002.

21. R. Echahed and F. Prost. Security policy in a declarative style. In *Proc. of PPDP'05*. ACM Press, 2005.
22. S. Escobar, C. Meadows and J. Meseguer. A Rewriting-Based Inference System for the NRL Protocol Analyzer and its Meta-Logical Properties. *Theoretical Computer Science*, Volume 367, Issues 1-2, pages 162-202, Elsevier 2006.
23. M. Fernández and J.-P. Jouannaud. Modular termination of term rewriting systems revisited. In *Proc. of ADT'94*, number 906 in LNCS, Santa Margherita, Italy, 1995.
24. R. Jagadeesan and V. Saraswat. Timed Constraint Programming: A Declarative Approach to Usage Control. In *Proc. of PPDP'05*. ACM Press, 2005.
25. S. Jajodia, P. Samarati, M. Sapino, and V.S. Subrahmaninan. Flexible support for multiple access control policies. *ACM TODS*, 26(2):214–260, 2001.
26. S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. *SIGMOD Rec.*, 26(2):474–485, 1997.
27. T. Jim. SD3: A trust management system with certified evaluation. In *IEEE Symp. Security and Privacy*, pages 106–115, 2001.
28. Jonscher, D. and Dittrich, K. An approach for building secure database federations. In Proc. of *VLDB'94*, ACM Trans. on Internet Technology, Vol. 8, No. 1, 1994.
29. J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, 17(1):4–23, 2005.
30. J.-W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems, introduction and survey. *TCS*, 121:279–308, 1993.
31. M. Koch, L. Mancini, and F. Parisi-Presicce. A graph based formalism for Rbac. In *SACMAT 2004*, pages 129–187, 2004.
32. N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *IEEE Symposium on Security and Privacy*, pages 114–130, 2002.
33. M.H.A. Newman. On theories with a combinatorial definition of equivalence. *Annals of Mathematics*, 43(2):223–243, 1942.
34. Department of Defense. Trusted computer system evaluation criteria, 1983. DoD 5200.28-STD.
35. A. Santana de Oliveira. *Réécriture et Modularité pour les Politiques de Sécurité*. PhD thesis, Université Henri Poincare, Nancy, France, 2008.