Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 234 (2009) 19–36
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Action Control by Term Rewriting*

Steve Barker[a] Clara Bertolissi[a,b] Maribel Fernández [a,1]

[a] *Dept. Computer Science, King's College London, Strand, WC2R 2LS, U.K*

[b] *LIF and Université de Provence, Marseille, France*

Abstract

We address the problem of defining access control policies that may be used in the evaluation of requests made by client actors, in the course of e-trading, to perform actions on the resources maintained by an e-collective. An e-collective is a group of agents that may act individually or in conjunction with other agents to satisfy a client's request to act. Our principal contribution to this key problem is to define formally an access control model in terms of which policies may be specified for helping to ensure that only legitimate forms of client actions are performed in the course of engaging in e-trading. We call this model the *action control model*. In action control, the notion of intentional, empowered, authorized actions, that may be performed individually or jointly with other agents and in a manner that is consistent with a group ethos, is the basis for specifying a set of *permissives*. A *permissive* is a generalization of the notion of permission (as the latter term is usually interpreted in access control). We define our action control model as a term rewrite system and we give examples of access policy representation.

Keywords: Access Control, Security, Term Rewriting.

1 Introduction

For many applications, existing access control models are sufficient for the limited forms of actions that are required on simple object types (e.g., read and write actions on files), which apply in the context of the restricted forms of organizations and systems that are often assumed (cf. the Bell-LaPadula model [6]). Nevertheless, there are organizational structures, emerging applications, new types of computer systems, as well as complex forms of actions, that require that existing access control models be extended and that novel access control models be developed to address access control policy requirements adequately.

In this paper, we address the issue of formally defining an access control model for use for secure e-trading where (pre-authenticated) client agents make requests for an action to be performed by a type of virtual organization that we call an

* Research partially funded by the EU project *Implementing access control mechanisms using rewriting techniques*, Marie Curie Intra European Fellowships Programme.

[1] Email: {Steve.Barker,Clara.Bertolissi,Maribel.Fernandez}@kcl.ac.uk

doi:10.1016/j.entcs.2009.02.070

e-collective. Informally, an e-collective is a collection of server agents that may act individually to satisfy a client's request that some action be performed or a subset of the agents in the e-collective may act jointly in order to satisfy the client's request. For example, given a client request to buy 1000 green widgets, members of the e-collective may offer to service the client's request in full or different subsets of the server agents of the e-collective may offer to satisfy part of the client's request. This paper addresses the problem of formally specifying which client actions can be performed by which server agents and in what ways.

For e-trading via e-collectives, a number of requirements exist. The client agents and server agents will generally be widely distributed, the e-trading environment will typically be highly dynamic (with trading conditions and policy requirements changing frequently), server agents will maintain individual policies that need to be combined to make collective decisions on access requests, and changes to policies will usually need to be performed autonomously. Traditional access control models often assume a single, monolithic policy specification, that is managed by human security administrators, that applies to a static, centralized organization with a well-defined set of users that request that simple forms of actions be performed on simple forms of objects. Such assumptions, and the access control models on which they are based, are inappropriate in the e-collective scenario.

For client request evaluation by e-collectives, server agent intentions and empowerments need to be considered, as well as authorizations. Put (very) simply, an intention is a will to act; an empowerment is a capacity to act. In the case of e-collectives, not all server agents will necessarily have the intention of satisfying all client requests for all actions to be performed on every resource at all times and, clearly, it will not be the case that all server agents will necessarily have the same capability of satisfying a client's request. More importantly, intentions, empowerments and authorizations are fundamentally and mutually dependent on whether the server agents act individually or collectively to satisfy a client agent's request. For instance, e-collective server agents might not be empowered to act individually to satisfy a client's request but may be empowered to act jointly with other servers to satisfy the request, not every server agent will necessarily have the intention of acting individually or engaging jointly to satisfy every client's request that some action be performed, and what a server agent authorizes may depend on whether it acts with other server agents or not. The importance of combining intentions, empowerments and authorizations for request evaluation for e-collectives has not hitherto been adequately considered in the literature on access control models. In addressing this shortcoming, we propose a new type of access control model, which we call the *action control model.*

For the formal definition of the action control model, we use *term rewriting* [1,12,19]. Term rewriting has a number of specific attractions for representing access control policies, as described in [5,8,29]. In particular, standard rewriting techniques and modularity results for rewriting systems can be used to show that access control policies satisfy essential properties (such as consistency and totality), and programming languages and tools such as ELAN [10,18], MAUDE [11], Haskell [16] and ML [26], can be used to test, compare and experiment with evaluation strategies, to automate equational reasoning, and for the rapid prototyping of policies. As we will see, action control generalizes access control. The extra complexities involved in specifying action control polices demand that high-level, declarative languages be used for policy specification, languages like those based on term rewriting. Moreover, recursive rule definition is particularly important in client request evaluation. For action control for e-collectives, the distributed term rewriting systems that we use are especially well-suited.

For the actions of relevance in e-trading (e.g., acts of *buying*, *supplying*, ...), we argue that action control requires that intentions and empowerments to act need to be considered as well as authorizations to act. The authorization of a requested action by the client agent becomes meaningful if and only if an intention to perform the action and an empowerment to so act exist for a server agent. Intentions and empowerments may be interpreted as preconditions for authorization. In the existing access control literature, a permission is a pair (a, r), where a is an action and r is a resource, and an authorization is a permission assigned to a client user. In contrast, in action control, a generalization of a permission is used, which we call a *permissive*. A permissive represents an optional, authorized, intentional, empowered action.

Permissives are, as we have said, fundamentally related to the mode in which a server agent acts, individually or collectively. When a server agent of an e-collective acts individually to satisfy a client's request then we say that the server agent acts in $\mathcal{I}$-mode; when an server agent must collaborate with other server agents in the e-collective to satisfy a client's request (with each server agent contributing to the satisfaction of the request) then we say that the server agent acts in $\mathcal{C}$-mode.[2] The distinction between $\mathcal{I}$-mode and $\mathcal{C}$-mode motivates the need for a shared meta-policy that determines how server agents may act individually or collectively to service client agent requests and to what extent. For that, the notion of a *group ethos* is used in action control. In the $\mathcal{C}$-mode case, a request by a client agent for an action to be performed may be satisfied by the server agents of the e-collective subdividing the request and satisfying part of the original request. In order for a permissive to hold in $\mathcal{I}$-mode or $\mathcal{C}$-mode the permissive must be consistent with the group ethos.

Despite being thus far relatively neglected, it is important for researchers to consider the range of access control issues that relate to e-collectives used for e-trading. For a number of reasons, if adequate access controls can be provided, e-collectives will become routinely used in, for example, grid computing, as a form of virtual organization, and in the context of Internet and Semantic Web technologies. As an e-collective, a group of server agents may be able to pool their resources to compete for a contract that none of them could individually satisfy, they may choose to subcontract a client's request for business reasons, and collections of server agents may decide to collaborate in order to combine their individual strengths and specializations.

[2] Our use of $\mathcal{I}$-mode and $\mathcal{C}$-mode actions is based on Tuomela's theory of cooperation [32].

The remainder of the paper is organized in the following way. Section 2 describes some formal preliminaries. In Section 3, we define our action control model as a term rewriting system and we give examples of policy usage and secure request evaluation. Section 4 discusses how some properties of action control policies, such as consistency, can be obtained using term rewriting techniques. In Section 5, we discuss the related literature more fully. In Section 6, conclusions are drawn and further work is suggested.

2 Preliminaries

In this section, we recall some basic notions and notations for term rewriting that we will use in the rest of the paper. We refer the reader to [1] for additional information.

A *signature* $\mathcal{F}$ is a finite set of *function symbols* together with their (fixed) arity. $\mathcal{X}$ denotes a denumerable set of *variables* $X_1, X_2, \ldots$, and $T(\mathcal{F}, \mathcal{X})$ denotes the set of *terms* built up from $\mathcal{F}$ and $\mathcal{X}$. Terms are identified with finite labeled trees. The symbol at the root of t is denoted by $root(t)$. *Positions* are strings of positive integers. 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 $t[u]_p$.

$\mathcal{V}(t)$ 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. We say that two terms unify if there is some substitution that makes them equal. Such a substitution is called a *unifier*. The *most general unifier* (mgu) is the unifier that will yield instances in the most general form.

Given a signature $\mathcal{F}$, a *term rewriting 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 2.1 Consider a signature for lists of natural numbers, with function symbols:

- z (with arity 0) and s (with arity 1, denoting the successor function) to build numbers;
- nil (with arity 0, to denote an empty list), cons (with arity 2, to construct non-empty lists), and length (with arity 1, to compute the length of a list).

The list containing the numbers 0 and 1 is written: $\mathsf{cons}(\mathsf{z}, \mathsf{cons}(\mathsf{s}(\mathsf{z}), \mathsf{nil}))$, or simply $[\mathsf{z}, \mathsf{s}(\mathsf{z})]$ for short. We can specify the function length with rewrite rules as follows:

$$\mathsf{length}(\mathsf{nil}) \to \mathsf{z} \qquad \mathsf{length}(\mathsf{cons}(X, L)) \to \mathsf{s}(\mathsf{length}(L))$$

Then we have a reduction sequence:

$$\mathsf{length}(\mathsf{cons}(\mathsf{z},\mathsf{cons}(\mathsf{s}(\mathsf{z}),\mathsf{nil}))) \to \mathsf{s}(\mathsf{length}(\mathsf{cons}(\mathsf{s}(\mathsf{z}),\mathsf{nil})) \to \mathsf{s}(\mathsf{s}(\mathsf{length}(\mathsf{nil}))) \to \mathsf{s}(\mathsf{s}(\mathsf{z}))$$

Example 2.2 Boolean operators, such as disjunction and conjunction, can be specified using a signature that includes two constants, true and false.

$$\begin{array}{ll} \mathsf{or}(\mathsf{true},X) \to \mathsf{true} & \mathsf{and}(\mathsf{true},X) \to X \\ \mathsf{or}(\mathsf{false},X) \to X & \mathsf{and}(\mathsf{false},X) \to \mathsf{false} \end{array}$$

In the sequel we use the notation t_1 and ... and t_n as syntactic sugar for the term $\mathsf{and}(\ldots\mathsf{and}(\mathsf{and}(t_1,t_2),t_3)\ldots)$, and similarly for the or operator.

Let $l \to r$ and $s \to t$ be two rewrite rules (we assume that the variables of $s \to t$ were renamed so that there is no common variable with $l \to r$), p the position of a non-variable subterm of s, and μ a most general unifier of $s|_p$ and l. Then $(t\mu, s\mu[r\mu]_p)$ is a *critical pair* formed from those rules. Note that $s \to t$ may be a renamed version of $l \to r$. In this case a superposition at the root position is not considered a critical pair. A term rewriting system R is:

- *confluent* if for all terms t, u, v: $t \to^* u$ and $t \to^* v$ implies $u \to^* s$ and $v \to^* s$, for some s;
- *terminating* (or *strongly normalizing*) if all reduction sequences are finite;
- *left-linear* if all left-hand sides of rules in R are linear;
- *non-overlapping* if there are no critical pairs;
- *orthogonal* if R is left-linear and non-overlapping;

For example, the rewrite system in Example 2.1 is left-linear and non-overlapping (therefore orthogonal), confluent and terminating.

Action control policies are applicable in distributed computing contexts. For that, we will use *distributed term rewriting systems (DTRSs)*; DTRSs are term rewriting systems where rules are partitioned into modules, each associated with a site, and function symbols are annotated with site identifiers. DTRSs were introduced in [8] to specify access control policies in a dynamic, distributed environment.

We assume that each site has a unique identifier (we use Greek letters $\mu, \nu, \ldots$ to denote site identifiers). We say that a rule $f(t_1,\ldots,t_n) \to r$ defines f. There may be several rules defining f; we write f_ν to refer to the definition of the function symbol f stored in the site ν. If a symbol is used in a rule without a site annotation then we assume the function is defined locally. For example, in a DTRS used in a bank scenario, we may have a local function account such that $\mathsf{account}(u)$ returns u's bank account number, and rules computing the average balance of a user's account, stored in a site ν. Then we could define the category of a user u by using a rule of the form

$$\begin{array}{l} \mathsf{category}(U) \to \mathit{if}\ \mathsf{averagebalance}_\nu(\mathsf{account}(U)) \geq 10000 \\ \qquad\qquad \mathit{then}\ \mathsf{VIP}\ \mathit{else}\ \mathsf{NORMALCLIENT} \end{array}$$

We use the notation *if* b *then* s *else* t as syntactic sugar for the term if-then-else(b, s, t), with the rewrite rules:

$$\text{if-then-else}(\mathsf{true}, X, Y) \to X \qquad \text{if-then-else}(\mathsf{false}, X, Y) \to Y$$

In this paper, we assume that the site where each function is defined is known and therefore the annotations used in function symbols are just constants. More general versions (where, for instance, variables can be used as annotations when sites are not known in advance), may also be used.

3 The Action Control Model by Term Rewriting

In this section, we formally define the action control model. We consider some basic syntactic issues, we then give and justify the inclusion of the core set of rewrite rules that define the action control model, we give examples of policy specification, in terms of the model, and we consider client request evaluation.

On syntax, we first note the key sets of constants in the signature that we use in the formulation of the action control model and policies. Specifically, we require:

- A countable set $\mathcal{C}$ of client identifiers, c_0, c_1, ...
- A countable set $\mathcal{M}$ of identifiers, $m_0, m_1, \ldots$ of server agents that are members of an e-collective.
- A countable set $\mathcal{A}$ of named *actions* (represented by strings that denote actions in a world of interest e.g., *buy*, *sell*, *supply*, *hire*, ...).
- A countable set $\mathcal{R}$ of *resource identifiers*, r, r_1, ...
- A countable set $\mathcal{T}$ of *time points*, t, t_1, ...

Here, we have that $\mathcal{C} \cap \mathcal{M} = \emptyset$ holds for a particular instance of an e-collective. We include times with our specifications of action control policies because intentions, empowerments, and authorizations will inevitably change, and often in a highly dynamic manner. We adopt a one-dimensional, linear, discrete view of time, with a beginning and no end point, i.e., a total ordering of time points that is isomorphic to the natural numbers. In the ensuing discussion, we represent times as encoded natural numbers in $YYYYMMDD$ format. We assume the existence of a function that extracts the current time from the system clock.

Recall that we are proposing an interpretation of access control in which intentions, empowerments and authorizations are necessary in order for a permissive to hold. It follows that an action control theory is a 4-tuple consisting of a set of permissives and three subtheories, in terms of which the set of permissives is defined,

$$(\mathcal{PER}, \mathcal{INT}, \mathcal{EMP}, \mathcal{AUTH}).$$

Here, $\mathcal{PER}$ is the set of rules that is used to define the set of permissives, $\mathcal{INT}$ is a theory of intentions that individual agents within the e-collective have to act (in $\mathcal{I}$-mode and $\mathcal{C}$-mode), $\mathcal{EMP}$ is a theory that specifies whether an agent in the e-collective is empowered to act according to the group ethos (in $\mathcal{I}$-mode or $\mathcal{C}$-mode), and $\mathcal{AUTH}$ is a theory that defines the actions that members of the e-collective authorize client agents to perform on resources (in $\mathcal{I}$-mode or $\mathcal{C}$-mode) that are maintained by the e-collective.

The examples of action control policy formulation that we will give, later in this section, relate to a simple e-commerce application. Specifically, we assume an e-collective with four members that are identified by m_1, m_2, m_3 and m_4. We make the simplifying assumptions that the only resource of interest to client agents is a set of *parts* and the only action of interest is an act of *buying*. Each member of an e-collective that has an intention of satisfying a client's request to buy some part will maintain its own information about the set of *parts* at its local site. We assume each part is identified by a name (widget, bauble, etc) and that client requests are to buy a quantity of a specified part. The binary constructor *part* is used to specify a part p and its quantity q.

Given an action control policy specification, a client c_0 may make a request of the form:

$$permissive(c_0, buy, part(widget, 1300), m_1).$$

That is, c_0's request is to know whether it is permitted to *buy* 1300 units worth of the *widget* part from the agent m_1. On receiving c_0's request, the agent m_1 will evaluate the request with respect to its own declarations of permissives and may request other e-collective members to cooperate with it to jointly satisfy c_0's request.

Our examples will also involve considering application-specific information about the stock levels of various parts held by the e-collective agents m_1, m_2, m_3 and m_4 and stored in sites identified by υ_1, υ_2, υ_3 and υ_4, respectively, viz:

$$\Big\{ stock_{\upsilon_1}(widget) \rightarrow 200,\ stock_{\upsilon_1}(bauble) \rightarrow 300, \ldots$$

$$\Big\{ stock_{\upsilon_2}(widget) \rightarrow 100,\ stock_{\upsilon_2}(bauble) \rightarrow 300, \ldots$$

$$\Big\{ stock_{\upsilon_3}(widget) \rightarrow 800,\ stock_{\upsilon_3}(bauble) \rightarrow 100, \ldots$$

$$\Big\{ stock_{\upsilon_4}(widget) \rightarrow 150,\ stock_{\upsilon_4}(bauble) \rightarrow 200, \ldots$$

Here, $stock_{\upsilon_i}(r) \rightarrow q$ iff the e-collective member m_i $(1 \leq i \leq 4)$ has the quantity q of the item of type r available in stock.[3]

3.1 $\mathcal{PER}$ Theory

The $\mathcal{PER}$ sub-theory consists of the following rules, which define permissives for each member m of the e-collective using a 4-ary function *permissive*. To specify the function *permissive* we use the functions $i\_permissive_\nu$ and $c\_permissive_\mu$, where ν, μ are the sites that store the action control policy for m, expressed in individual and collective mode, respectively. We define $i\_permissive_\nu$, $c\_permissive_\mu$ and

[3] *We assume that the stock information used by different e-collective members is of the same format, but this need not, of course, be the case.*

permissive for each agent in the e-collective. These core rules are defined in the following way:

$$\begin{aligned}
permissive(C, A, R, m) &\rightarrow i\_permissive_{\nu}(C, A, R) \text{ or } c\_permissive_{\mu}(C, A, R)\\
i\_permissive_{\nu}(C, A, R) &\rightarrow F(i\_empowered_{\nu}(C, A, R), i\_intent_{\nu}(C, A, R),\\
&\qquad i\_authorized_{\nu}(C, A, R))\\
c\_permissive_{\mu}(C, A, R) &\rightarrow G(c\_empowered_{\mu}(C, A, R), c\_intent_{\mu}(C, A, R),\\
&\qquad c\_authorized_{\mu}(C, A, R))
\end{aligned}$$

According to the rules above, when a client c requests a member m of the e-collective to perform the action a on resource r, the agent m has the option to act in $\mathcal{I}$-mode or $\mathcal{C}$-mode to satisfy the request. In the rules above, $i\_permissive_{\nu}$, i_intent_{ν} and $i\_authorized_{\nu}$ are functions returning boolean values. The functions F and G are parameters of the specification. They may simply be a conjunction operator, or we may have additional conditions that need to be verified in order to define $i\_permissive_{\nu}$ and $c\_permissive_{\mu}$. If we assume that F and G are simply computing a conjunction, then the rule defining *i_permissive* specifies that client c's request to perform the a action on resource r is satisfied by the e-collective agent m acting in $\mathcal{I}$-mode if m has the intention of performing the action a on the resource r, in $\mathcal{I}$-mode, m is empowered, in $\mathcal{I}$-mode, to perform the action a on r, and m authorizes the client c to perform the action a in $\mathcal{I}$-mode. Thus, we might have, for example, the following reduction:

$$\begin{aligned}
i\_permissive_{\nu}(c, a, r) &\rightarrow \mathsf{and}(i\_empowered_{\nu}(c, a, r), i\_intent_{\nu}(c, a, r),\\
&\qquad i\_authorized_{\nu}(c, a, r))\\
&\rightarrow \mathsf{and}(\mathsf{true}, \mathsf{true}, \mathsf{true}) \rightarrow \mathsf{true}
\end{aligned}$$

Similarly, the *c_permissive* rule specifies that client c's request to perform the a action on resource r is satisfied by the e-collective agent m acting in $\mathcal{C}$-mode if m has the intention of performing the action a on the resource r, in $\mathcal{C}$-mode, m is empowered, in $\mathcal{C}$-mode, to perform the action a on r, and m authorizes c, in $\mathcal{C}$-mode, c's requested action. We consider the representation of theories of intentions, empowerments and authorizations next.

3.2 $\mathcal{INT}$ Theory

Each server agent of the e-collective will maintain its own intention theory $\mathcal{INT}$. $\mathcal{INT}$ is a set of rules that define agent m's intentions in terms of an action a that m is willing to engage in on resource r, in $\mathcal{I}$-mode, on client c's behalf (the *i_intent* rules), and m's intentions in terms of an action a that m is willing to engage in on resource r, in $\mathcal{C}$-mode, on c's behalf (the *c_intent* rules):

$$i\_intent(C, A, R) \rightarrow C_1 \textsf{ and } \dots \textsf{ and } C_n$$

$$c\_intent(C, A, R) \rightarrow C_1 \textsf{ and } \dots \textsf{ and } C_k$$

The rules above (where site annotations have been omitted) specify that the e-collective agent m has an intention to perform an action a on resource r in $\mathcal{I}$-mode ($\mathcal{C}$-mode) on behalf of client c if the conditions, $C_1\sigma, \dots, C_n\sigma$ ($C_1\sigma, \dots, C_k\sigma$) hold, where $\sigma = \{C \mapsto c, A \mapsto a, R \mapsto r\}$. The example that follows next illustrates what is involved in specifying server agent intentions as a term rewriting systems.

Example 3.1 Consider the following set of intention theories $\mathcal{INT}_{m_1}$, $\mathcal{INT}_{m_2}$, $\mathcal{INT}_{m_3}$, and $\mathcal{INT}_{m_4}$, for the e-collective members m_1, m_2, m_3 and m_4:

$$\mathcal{INT}_{m_1} := \begin{cases} i\_intent_\nu((C, buy, part(X, Z)) \rightarrow \textsf{true} \\ c\_intent_\mu((C, buy, part(X, Z)) \rightarrow \ Z > 1000 \end{cases}$$

$$\mathcal{INT}_{m_2} := \begin{cases} i\_intent_\alpha(C, buy, part(widget, Z)) \rightarrow Z \leq 1000 \\ c\_intent_\tau(C, buy, part(bauble, Z)) \rightarrow Z \geq 50 \end{cases}$$

$$\mathcal{INT}_{m_3} := \Big\{ i\_intent_\pi(C, A, R) \rightarrow preferred(C)$$

$$\mathcal{INT}_{m_4} := \begin{cases} i\_intent_\gamma(C, A, R) \rightarrow current\_time \geq 20080601 \\ c\_intent_\delta(C, A, R) \rightarrow current\_time \geq 20081001 \end{cases}$$

Here, $\mathcal{INT}_{m_1}$ is a specification of the intentions that the e-collective member m_1 has of satisfying any client's request to perform an act of buying in $\mathcal{I}$-mode or in $\mathcal{C}$-mode provided that the total size of the order (in the latter case) is greater than 1000 units. $\mathcal{INT}_{m_2}$ expresses m_2's intention to act individually to satisfy a request from any client to buy if the request is for less than 1000 widgets; m_2 is also willing to act in $\mathcal{C}$-mode to satisfy requests for sales of more than 50 units worth of baubles. $\mathcal{INT}_{m_3}$ expresses that m_3 is willing to satisfy any action on any resource without restriction, but only in $\mathcal{I}$-mode and only for the clients that it identifies as *preferred*. Finally, $\mathcal{INT}_{m_4}$ specifies that m_4 is willing to satisfy any action requested by any client on any resource in $\mathcal{I}$-mode as soon as m_4 starts trading on 1st June 2008 and will act in $\mathcal{C}$-mode four months after starting to trade.

3.3 $\mathcal{EMP}$ Theory

In addition to its intentions, each server member of the e-collective will maintain its own $\mathcal{EMP}$ theory, a specification of its empowerments. An $\mathcal{EMP}$ theory, is a set of rules of the form:

$$i\_empowered(C, A, R) \rightarrow C_1 \textsf{ and } \dots \textsf{ and } C_n$$

$$c\_empowered(C, A, R) \rightarrow C_1 \textsf{ and } \dots \textsf{ and } C_k$$

Each $i\_empowered$ ($c\_empowered$) rule specifies that an agent m is empowered to perform the a action on resource r for client c in $\mathcal{I}$-mode ($\mathcal{C}$-mode) if the conjunction of conditions $C_1, \ldots, C_n$ ($C_1, \ldots, C_k$) hold. In the case of $\mathcal{C}$-mode evaluation, at least one C_i ($1 \leq i \leq k$) should be expressed in terms of $c\_permissive$. The reason for this should be noted: in the case of $\mathcal{C}$-mode evaluation, the evaluation of $c\_empowered(C, A, R)$ involves a member m of the e-collective requesting the collaboration of other agents in the e-collective in satisfying the request that an a action be performed in relation to resource r on behalf of the client c. Hence, a $c\_permissive$ condition in a $c\_empowered$ rule is a recursive call that will trigger the evaluation of a request by a set of members of the e-collective, that will collaborate with m to satisfy the request by c to perform the action a on r. For example, we might have a reduction:

$$c\_empowered_{\mu}(c, a, r) \rightarrow c_1(a) \text{ and } c\_permissive_{\tau}(c, a, r) \rightarrow \text{true and true} \rightarrow \text{true}$$

Hence, when m acts in $\mathcal{C}$-mode according to the policy specified in site μ, then m acts jointly with other e-collective members if and only if there is some e-collective agent $m' \neq m$ (whose action control policy is stored in site τ) that has an intention and is empowered and authorized to act with m to satisfy a client's request.

To treat adequately the concept of empowerment in action control for e-collectives, the notion of a group ethos is required. The group ethos determines what powers member agents can exercise in the context of the e-collective. To aid the reader's understanding of this key issue, we describe a simple form of group ethos such that an e-collective member m acts to satisfy a request from a client c in full by m operating alone whenever that is consistent with m's specifications of intentions, empowerments and authorizations. If m has an intention to act but is not empowered to act then m will request other agents of the e-collective to help in satisfying c's request. Hence, each member of the e-collective will act in $\mathcal{I}$-mode if possible and in $\mathcal{C}$-mode otherwise. Despite its simplicity, this group ethos is consistent with the concept of *prospective rationality* that is defined within the framework of *Rational Choice* [22]. Henceforth, we will call the shared ethos such that e-collective members act to their maximum intended, empowered and authorized extent to satisfy a client's request in $\mathcal{I}$-mode, whenever that is possible, the *Principle of Maximal Individualistic Satisfaction*, to wit:

If an e-collective member m intends and can fully satisfy a request of a client c for an action to be performed that m authorizes for c then m will act in an $\mathcal{I}$-mode capacity (i.e., privately, selfishly) subject to any constraints on $\mathcal{I}$-mode action (conversely, m has the minimal commitment to act collectively). Otherwise, m determines the maximal extent to which it can satisfy the request by c, and will request help from other members of the e-collective on how to satisfy the remainder of c's request. In this case, the $\mathcal{C}$-mode case, m invites members of the e-collective to engage in joint activity to satisfy c's request.

Acting as a member agent entails respecting the Principle of Maximal Individualistic Satisfaction, the shared group ethos. Other forms of group ethos are, of course, possible (e.g., to allow for group competitiveness, altruism, etc); these alternatives are the basis for different e-collective meta-policies that can, nevertheless, be naturally represented in our framework.

The next example illustrates how empowerment theories may be represented as a TRS.

Example 3.2 Consider the following empowerment theories $\mathcal{EMP}_{m_1}$, $\mathcal{EMP}_{m_2}$, $\mathcal{EMP}_{m_3}$, and $\mathcal{EMP}_{m_4}$, for the e-collective members m_1, m_2, m_3 and m_4, respectively. The empowerment theories assume that the Principle of Maximal Satisfaction is to be implemented.

$$\mathcal{EMP}_{m_1} := \begin{cases} i\_empowered_{\nu}(C, buy, part(X, Z)) \rightarrow stock_{v_1}(X) \geq Z \\ c\_empowered_{\mu}(C, buy, part(X, Z)) \rightarrow stock_{v_1}(X) < Z \text{ and} \\ \qquad c\_permissive_{\tau}(C, buy, part(X, Z - stock_{v_1}(X))) \end{cases}$$

$$\mathcal{EMP}_{m_2} := \begin{cases} i\_empowered_{\alpha}(C, buy, part(X, Z)) \rightarrow \\ \qquad month(current\_time) > 6 \text{ and} \\ \qquad stock_{v_2}(X) > Z \text{ and } Z \leq 500 \\ c\_empowered_{\tau}(C, buy, part(X, Z)) \rightarrow Z \geq 500 \text{ and} \\ \qquad c\_permissive_{\delta}(C, buy, part(X, Z - stock_{v_2}(X))) \end{cases}$$

$$\mathcal{EMP}_{m_3} := \Big\{ i\_empowered_{\pi}(c_0, buy, part(X, Z)) \rightarrow stock_{v_3}(X) \geq Z$$

$$\mathcal{EMP}_{m_4} := \begin{cases} i\_empowered_{\gamma}(C, buy, part(X, Z)) \rightarrow stock_{v_4}(X) \geq Z \\ c\_empowered_{\delta}(C, buy, part(X, Z)) \rightarrow stock_{v_4}(X) < Z \text{ and} \\ \qquad c\_permissive_{\mu}(C, buy, part(X, Z - stock_{v_4}(X))) \end{cases}$$

Here, the agent m_1 will act in $\mathcal{I}$-mode for any client c if m_1 is empowered to so act by having enough stock, as recorded in v_1, to satisfy c's request to perform an act of buying. Alternatively, if m_1 cannot satisfy the request itself then m_1 will attempt to act in $\mathcal{C}$-mode, with another member of the e-collective, to service c's request. That is, m_1 will adhere to the Principle of Maximal Individualistic Satisfaction and satisfy whatever it can of c's request and then ask for assistance with the remainder. In contrast, m_2 is only empowered to act in $\mathcal{I}$-mode if it has enough stock as recorded in v_2, subject to certain temporal constraints being satisfied, and for orders of less than 500 units. In the case where a request is to buy more than 500 units, m_2 will act in $\mathcal{C}$-mode but by requesting help from δ, to which m_2 subcontracts "large" orders. Furthermore, while m_2 is willing to supply any client, m_3 is only empowered to supply c_0 and in $\mathcal{I}$-mode. Similarly to m_1, m_4 is only empowered to act in $\mathcal{I}$-mode if it has enough stock, otherwise it will act in $\mathcal{C}$-mode and ask for the collaboration of another member of the e-collective for satisfying the client's request.

3.4 $\mathcal{AUTH}$ Theory

In addition to an $\mathcal{INT}$ theory and an $\mathcal{EMP}$ theory, each agent m in the e-collective will define an $\mathcal{AUTH}$ theory. An $\mathcal{AUTH}$ theory is a set of rules for specifying whether m authorizes a client c to perform an action a on a resource r when m is acting in $\mathcal{I}$-mode or $\mathcal{C}$-mode. The specification of the authorization policy may follow any of the standard access control models. For instance, we may specify a role-based authorization policy. We omit the details of the access control model (see for instance [5] for a rewrite-based specification of the RBAC model) and just show the general form of the rules:

$$i\_authorized(C, A, R) \rightarrow \rho_i$$

$$c\_authorized(C, A, R) \rightarrow \rho_c$$

where ρ_i and ρ_c are suitable right-hand sides (depending on the policy to be modeled). Therefore, the term $i\_authorized(c, a, r)$ (or $c\_authorized(c, a, r)$) is evaluated using the rewrite rules defining the chosen access control policy and gives as a result a permission or a denial, denoted by true and false respectively, for client agent c's request to perform an action a on resource r.

An example of an $\mathcal{AUTH}$ subtheory, represented as a TRS, follows next.

Example 3.3 For the authorization subtheory that is used in our running example, we make the simplifying assumption that all members of the e-collective use the same RBAC authorization theory, which includes the following definitions:

$$i\_authorized(C, A, R) \rightarrow \mathsf{check}(\mathsf{member}((A, R), \mathsf{privileges}(\mathsf{roles}(C))))$$

$$c\_authorized(C, A, R) \rightarrow \mathsf{check}(\mathsf{member}((A, R), \mathsf{privileges}(\mathsf{roles}(C))))$$

$$\begin{array}{llll} \mathsf{roles}(c_0) \rightarrow [r_{01}, \ldots, r_{1i}] & \ldots & \mathsf{roles}(c_n) \rightarrow [r_{n1}, \ldots, r_{nk}] \\ \mathsf{priv}(r_0) \rightarrow [(a_{01}, o_{11}), \ldots, (a_{1i}, o_{1i})] & \ldots & \mathsf{priv}(r_n) \rightarrow [(a_{n1}, o_{n1}), \ldots, (a_{nk}, o_{nk})] \end{array}$$

$$\mathsf{privileges}(\mathsf{nil}) \rightarrow \mathsf{nil}$$

$$\mathsf{privileges}(\mathsf{cons}(R, L)) \rightarrow \mathsf{priv}(R) \cup \mathsf{privileges}(L)$$

In this example, a client agent C is authorized by an e-collective member to perform the A action on resource R, in $\mathcal{I}$-mode or $\mathcal{C}$-mode, if C is assigned to a role R' and the A privilege on R is assigned to R'. For example, given the rules
$\mathsf{roles}(c_0) \rightarrow [specialClient]$
$\mathsf{priv}(specialClient) \rightarrow [(buy, part(widget, 1300)), \ldots]$

we have

$$i\_authorized(c_0, buy, part(widget, 1300)) \rightarrow$$

$$\mathsf{check}(\mathsf{member}((buy, part(widget, 1300)), \mathsf{privileges}(\mathsf{roles}(c_0)))) \rightarrow$$

$$\mathsf{check}(\mathsf{member}((buy, part(widget, 1300)), \mathsf{privileges}([specialClient]))) \rightarrow^* \mathsf{true}$$

Any number of authorization theories can, of course, be defined (for the $\mathcal{I}$-mode and the $\mathcal{C}$-mode cases), but the simple authorization theory above is sufficient for understanding what is involved in specifying $\mathcal{AUTH}$ subtheories in action control.

3.5 Client Request Evaluation

We show next an evaluation example, using the setting and the theories presented in Examples 3.1, 3.2, and 3.3.

Example 3.4 Let us consider the request $permissive(c_0, buy, part(widget, 1300), m_1)$. From the previous discussion and using the examples of policy formulation that we have developed, we have the reduction

$$permissive(c_0, buy, part(widget, 1300), m_1) \rightarrow$$

$$i\_permissive_\nu(c_0, buy, part(widget, 1300)) \ \mathsf{or}\ c\_permissive_\mu(c_0, buy, part(widget, 1300))$$

The first term in the disjunction, corresponding to an action performed in $\mathcal{I}$-mode, leads to the result

$$\mathsf{and}(i\_empowered_\nu(c_0, buy, part(widget, 1300)), i\_intent_\nu(c_0, buy, part(widget, 1300)),$$

$$i\_authorized_\nu(c_0, buy, part(widget, 1300))) \ \rightarrow^* \ \mathsf{and}(\mathsf{false}, \mathsf{true}, \mathsf{true}) \ \rightarrow \ \mathsf{false}$$

since the stock provision (200 widgets) associated with member m_1 is not sufficient to satisfy the client's request.

The second term in the disjunction, corresponding to an action performed in $\mathcal{C}$-mode, leads instead to the result

$$\mathsf{and}(c\_empowered_\mu(c_0, buy, part(widget, 1300)), c\_intent_\mu(c_0, buy, part(widget, 1300)),$$

$$c\_authorized_\mu(c_0, buy, part(widget, 1300))) \ \rightarrow^* \ \mathsf{and}(\mathsf{true}, \mathsf{true}, \mathsf{true}) \ \rightarrow \ \mathsf{true}$$

assuming $c\_permissive_\tau(c_0, buy, part(widget, 1100))$ holds, i.e., another member at site τ is able to provide the remaining 1100 widgets. Therefore

$$permissive(c_0, buy, part(widget, 300), m_1) \rightarrow \mathsf{true}$$

In conclusion, the client's demand cannot be treated by member m_1 in isolation, but can be satisfied if other members of the e-collective collaborate with member m_1, in a way allowed by the action control policy.

4 Properties of Action Control Policies

As in the case of access control, action control policies should satisfy certain criteria in order to be "acceptable". For instance, it may be necessary to ensure that an action control policy formulation does not specify that any client is granted and denied the same request to the same agent at the same moment in time (i.e., that the policy is consistent). More precisely, we are interested in the following properties of action control policies:

Totality: Each request from a valid client c to a valid agent in the e-collective to perform a valid action a on the resource r receives as answer. Answers can be: a permission, a denial, or undeterminate (i.e., any result that is neither true nor false will be interpreted as undeterminate).

Consistency: For any $c \in \mathcal{C}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$, and $m \in \mathcal{M}$ it is not possible to derive more than one result for a request from c to m to perform the action a on r. In other words, at most one of the results true, false or undeterminate is possible for each request.

Totality and consistency can be ensured, for policies defined as term rewriting systems, by checking that the rewrite relation is confluent and terminating. Termination ensures that all requests produce a result (their evaluation cannot get "stuck") and confluence ensures that results are unique.

A stronger version of totality, requiring that the policy always produces an answer which is true or false (i.e., no undeterminate results) is desirable for some applications. In order to prove that a policy is total in this sense, a characterisation of normal forms is needed. Sufficient completeness is a useful property in this case (see [29]).

Termination and confluence of term rewriting systems are undecidable properties in general, but there is an extensive body of work aimed at providing decidable sufficient conditions for confluence and termination (see, for example, [20,25,24,3,2,9]). In particular, orthogonal systems are confluent, and systems that define recursive functions using a primitive recursive scheme are terminating. These conditions have been used in [5] to derive sufficient conditions for totality and confluence of access control policies, and the same techniques can be used for action control policies.

For example, it is easy to show that the rewrite system defined by the rules given in the examples in the previous section is confluent (even if some of the rules are not completely specified) because all left-hand sides are left-linear and non-overlapping (i.e., the rewrite system is orthogonal [20]). As a consequence:

Proposition 4.1 *The action control policy for the e-commerce application specified by the rewrite rules in Section 3 is consistent.*

5 Related Work

Access control requirements for e-trading via e-collectives are not well served by existing models. Traditional access control models, like *access control matrices* [21], are only suitable as authorization models for actions performed by individual agents within the context of static systems; they are not appropriate in the case of dynamic, e-trading via e-collectives.

In [27], an approach for controlled collaboration via the editing of data structures is proposed. The work is based on notions from RBAC and a collection of “collaboration rights”. In contrast, the action control model focuses on arbitrary actions and a conception of collaboration that is very different from that described in [27]. In [33], some primitive concepts that relate to the authorization of joint actions are discussed and are of relevance to the notion of cooperative action that we have considered. However, the work in [33] is based on a different conception of collective action than the one that we adopt and, unlike action control, there is no discussion of any formal aspects of joint action and no formal security model is proposed.

In principle, standard RBAC models [14] *do*, permit requested actions on resources to be of arbitrary complexity, but the support for arbitrariness is facilitated by RBAC’s (generally) restricted notion of authorization. In the standard definition of authorization in RBAC (omitting considerations of sessions), an action a is allowed on resource r iff a client c is assigned to a role to which the permission (a, r) has been assigned. In RBAC, a can be of arbitrary complexity and so the definitions of authorizations can be used to evaluate “Can I perform action a?” requests where a “yes” answer is rendered if c is authorized to perform a. However, for actions like *buying*, a “yes” response requires more than simply an authorization; a “yes” response for an act of *buying* only makes sense if there is a seller that has the intention of allowing c to buy and the power to sell to c. In action control, permissives are a generalization of the concept of authorization.

Work on *Context-TBAC (C-TBAC)* [15] is related to our action control model, it being concerned with access control in collaborative environments and the context in which access control requirements are defined. However, in the action control model, unlike C-TBAC, intention policies and empowerment policies are fundamentally important and are used as standard types of contexts. Work on TeaM-based Access Control [30] is also related to ours in the sense that collaborative activity may be engaged in by a “team” of agents to satisfy some goal (i.e., $\mathcal{C}$-mode action). However, the work on TMAC has been largely based on integrating TMAC with RBAC. More recently, a *locale-based access control (LBAC) model* [31] has been described that exhibits similarities to action control. However, LBAC is concerned with an specific enhancement of RBAC to take into account the context in which roles may be activated. None of C-TBAC, TMAC and LBAC make the distinction between $\mathcal{I}$-mode and $\mathcal{C}$-mode evaluation, identify the importance of a group ethos or use permissives for the rich forms of access control policies that are required in e-trading via e-collectives.

On the issue of context in access control, we also note that there are several contributions to the literature on representing contexts via rule-based specifications of conditional access control requirements (see, for example, [7], [17] and [4]). Moreover, XACML [23] may be used to express some quite general requirements for access control. In contrast to these approaches, the action control model is very specific about the types of rules that are to be admitted for defining permissives; *ad hoc* rules for specifying intentions, empowerments and authorizations are not simply to be included in policy specifications (for e-collectives) as “side-conditions”. Moreover, it is the emphasis on individual and collective acting in e-collectives that makes intentions, empowerments and a group ethos important, as well as authorizations, for action control. Rule-based approaches have thus far been not applied in cases where these notions emerge as being important to treat.

On access control by term rewriting, we note that [5,28,8] use term rewrite rules to represent access control policies. Our work differs from this literature in the sense that we have used term rewriting to define a new type of access control model, the action control model, which is intended for use for access control for a type of organization (e-collectives) to which standard assumptions do not apply (e.g., the assumption of centralized organizations to which static forms of access policies relate to simple forms of actions on simple types of objects). The work on “policy composition” by term rewriting by Dougherty et al [13] bears some similarity to ours. In Dougherty et al.’s work, the problem of resolving conflicts when combining access control policies is addressed. In contrast, our concern is with combining action control policies that define intentions, empowerments and authorizations and thus permissives that are used to define allowed forms of action requests.

6 Conclusions and Further Work

We have addressed the (open) problem of defining permitted forms of complex, requested actions (like *buying*) that are used by agents acting individually or collaboratively as part of an e-collective to evaluate client user requests. On that, our principal contribution has been to formally define an action control model in terms of which permissives may be specified: intentional, empowered authorized actions that may be performed individually (in $\mathcal{I}$-mode) or jointly (in $\mathcal{C}$-mode) and that are consistent with the shared ethos of an e-collective. To the best of our knowledge, no access control model has thus far been proposed that combines these concepts, that grounds them in an adequate theory of cooperation (cf. our use of [32]), and that is formally well-defined (cf. our use of term rewriting). For the formal specification, we have defined our action control model as a term rewrite system. Represented as a TRS, the action control model: enables changes to access control policies to be effected dynamically and autonomously; makes it possible to treat individualistic and joint actions and the combining of multiple e-collectives in a completely uniform manner (in both cases, by exploiting the recursion, which TRSs provide); and permits properties of policies to be defined and proven for assurance purposes. Moreover, despite the complexities involved in formulation, action control policies can be straightforwardly specified as a TRS.

To simplify the discussion in this paper, we have described the action control model in terms of a basic form of group ethos and we have used simple intention, empowerment and authorization theories in our examples. It must be noted, however, that many forms of ethos and more complex forms of action control policies can be naturally accommodated in the action control model and can be represented as term rewrite systems. In future work, we plan to consider other forms of group ethos and their representation as TRSs; we also intend to consider an enhanced form of the action control model on which constraints may be expressed to capture higher-level access control policy requirements.

References

[1] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, Great Britain, 1998.

[2] S. van Bakel and M. Fernández. Normalization results for typeable rewrite systems. *Information and Computation*, 133(2):73–116, 1997.

[3] F. Barbanera and M. Fernández. Intersection type assignment systems with higher-order algebraic rewriting. *Theoretical Computer Science*, 170:173–207, 1996.

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

[5] Steve Barker and Maribel Fernández. Term rewriting for access control. In *DBSec*, pages 179–193, 2006.

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

[7] Elisa Bertino, Barbara Catania, Elena Ferrari, and Paolo Perlasca. A logical framework for reasoning about access control models. In *SACMAT*, pages 41–52, 2001.

[8] Clara Bertolissi, Maribel Fernández, and Steve Barker. Dynamic event-based access control as term rewriting. In *DBSec*, pages 195–210, 2007.

[9] F. Blanqui, J.-P. Jouannaud, and M. Okada. Inductive data type systems. *Theoretical Computer Science*, 272(1-2):41–68, 2002.

[10] P. Borovansky, C. Kirchner, H. Kirchner, and P-E. Moreau. ELAN from a rewriting logic point of view. *Theoretical Computer Science*, 285:155–185, 2002.

[11] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. The Maude 2.0 system. In *Rewriting Techniques and Applications (RTA 2003)*, number 2706 in Lecture Notes in Computer Science, pages 76–87. Springer-Verlag, 2003.

[12] N. Dershowitz and J.-P. Jouannaud. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Formal Methods and Semantics*, volume B. North-Holland, 1989.

[13] Daniel J. Dougherty, Claude Kirchner, Hélène Kirchner, and Anderson Santana de Oliveira. Modular access control via strategic rewriting. In *ESORICS*, pages 578–593, 2007.

[14] David F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[15] Christos K. Georgiadis, Ioannis Mavridis, George Pangalos, and Roshan K. Thomas. Flexible team-based access control using contexts. In *SACMAT*, pages 21–27, 2001.

[16] Paul Hudak, John Peterson, and Joseph Fasel. A gentle introduction to Haskell 98. http://www.haskell.org/tutorial/, 1999.

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

[18] C. Kirchner, H. Kirchner, and M. Vittek. *ELAN user manual.* Nancy (France), 1995. Technical Report 95-R-342, CRIN.

[19] J.-W. Klop. Term Rewriting Systems. In S. Abramsky, Dov.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2. Oxford University Press, 1992.

[20] J.-W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems, introduction and survey. *Theoretical Computer Science*, 121:279–308, 1993.

[21] Butler W. Lampson. Protection. *SIGOPS Oper. Syst. Rev.*, 8(1):18–24, 1974.

[22] Tadao Miyakawa. *The Science of Public Policy.* Routledge, 1999.

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

[24] V. vanOostrom. Confluence for Abstract Higher-Order Rewriting. Free University of Amsterdam, 1994.

[25] Femke van Raamsdonk. Confluence and Normalisation for Higher-Order Rewriting, Free University of Amsterdam, 1996.

[26] D. Sannella S. Kahrs and A. Tarlecki. The definition of Extended ML: A gentle introduction. *Theoretical Computer Science*, 173(2):445–484, 1997.

[27] HongHai Shen and Prasun Dewan. Access control for collaborative environments. In *Proc ACM Conf. Computer-Supported Cooperative Work CSCW*, pages 51–58, 1992.

[28] A. Santana de Oliveira. Rewriting-based access control policies. *Proc. of SECRET'06, ENTCS*, 2006.

[29] A. Santana de Oliveira. Rewriting and modularity of security policies. PhD Thesis, University Henri Poincaré, Nancy, 2008.

[30] Roshan Thomas. Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments. In *ACM Workshop on Role-Based Access Control*, pages 13–19, 1997.

[31] W. Tolone, R. Gandhi, and G. Ahn. Locale-based access control: placing collaborative authorization decisions in context. In *Proc IEEE Conf. Systems, Man and Cybernetics*, 2003.

[32] R. Tuomela. *Cooperation.* Kluwer, 1999.

[33] Vijay Varadharajan and Phillip Allen. Support for joint action based security policies. In *ACISP*, pages 207–218, 1996.