Socially Constructed Trust for Distributed Authorization

Steve Barker[1] and Valerio Genovese*[2,3]

[1] King's College London, UK
[2] University of Torino, Italy
[3] University of Luxembourg, Luxembourg

Abstract. We describe an approach for distributed access control that is based on the idea of using a community-constructed repository of expressions of propositional attitudes. We call this repository an *oracle*. Members of a community may consult the oracle and use the expressions of belief and disbelief in propositions that are expressed by community members about requesters for access to resources. Our conceptual model and access control policies are described in terms of a computational logic and we describe an implementation of the approach that we advocate.

1 Introduction

An important aspect that makes distributed access control different to centralized scenarios, is that a reference monitor may have insufficient knowledge of requesters to decide whether to grant or deny access to a guarded resource. Hence, the reference monitor may need to "complete" its knowledge by relying on the testimony supplied to it by one or more trusted third parties.

Logic has had a prominent role to play in the specification, reasoning and enforcement of access control policies in which testimonial knowledge is distributed among several peers (also called principals). In particular, logic programming has been used for specifying policies and for implementing primitives to aggregate principals' knowledge.[1] However, such primitives present serious challenges when dealing with real-world, highly-distributed, large-scale and open access control scenarios like e-trading systems, social networks or microdata publishing platforms (e.g., Twitter). In such cases, the reference monitor may need to remotely access the testimonial knowledge of *large communities* of principals and this knowledge may be *temporarily missing* (e.g., some remote source is not available) or even *conflicting* (e.g., two principals in the same community provide contradictory information). Unfortunately, existing access control primitives to aggregate distributed knowledge do not scale to large communities of principals and provide limited support for dealing with incomplete or conflicting information. In this paper we address the following research question: *How to define an access control framework in which testimonial knowledge results from the aggregating of information issued by communities of principals?*

* Valerio Genovese is supported by the National Research Fund, Luxembourg.

[1] Examples of such primitives are delegation structures [18], boolean principals [1], dynamic or static thresholds [23, 17]

This main question reduces to a number of subquestions: How to aggregate testimony and reason with individual and aggregated testimony? How to define and implement operators to query remote knowledge bases? How to deal with conflicting information that stems from the aggregation of principals' testimonial warrant? How to define a specification language that admits nonmonotonic access control policy formulation? What operational methods can be used to evaluate access requests in a correct and computationally viable manner?

In answering these questions we describe an architecture that is based on the idea of using logical databases called "oracles" that speak-for a community on matters relating to requesters of access to resources. We present an access control framework based on Answer Set Programming (ASP) which extends the meta-model of access control introduced in [4] in several ways. We introduce primitives to accommodate the community-based view that we propose. In particular, we take testimonial knowledge as being expressed in the form of *propositional attitudes reports* of type "principal A believes ψ" or "principal A disbelieves ψ". Such testimonial knowledge is collectively stored by oracles. Finally, we extend the ASP system DLV [16] to include scalable primitives for querying several different remote knowledge bases during policy evaluation.

The remainder of the paper is organized as follows. In Section 2, we describe some details on which our policy specification language is based. In Section 3, we describe the architecture of our proposed system and the types of rules we allow for policy specification. In Section 4, we describe examples of the use of access control policy specifications by "oracles" and, in Section 5, we describe examples of the use of specifications of access control policies by reference monitors (called "acceptors") that make use of (or choose to accept) the policy information that is maintained by oracles. In Section 6, we describe some practical issues of relevance to our approach. In Section 7, we describe related work and, in Section 8, we draw conclusions and suggest further work.

2 A Community Security Language

In this section, we introduce the main syntactic and semantic concepts of our $\mathcal{C}$ommunity $\mathcal{S}$ecurity $\mathcal{L}$anguage ($\mathcal{CSL}$). The main sorts of constants for $\mathcal{CSL}$ are: a countable set $\mathcal{C}$ of categories, where c_0, c_1, ... are (strings) that are used to denote arbitrary category identifiers; a countable set $\mathcal{P}$ of principals, where p_0, p_1, ... are used to identify users, organizations, processes, ...; a countable set $\Sigma \subseteq \mathcal{P}$ of *sources* of testimonial knowledge, where s_0, s_1, ... are (strings) used to denote arbitrary sources of testimony; a countable set $\mathcal{A}$ of named atomic *actions*, where a_0, a_1, ... are (strings) used to denote arbitrary action identifiers; and a countable set $\mathcal{R}$ of *resource identifiers*, where r_0, r_1, ... denote arbitrary resources; $r(t_1, \ldots, t_n)$ is an arbitrary n-place relation that represents an "information resource" where t_i $(1 \leq i \leq n)$ is a term (a term is a function, a constant or a variable).

Informally, a category is any of several fundamental and distinct classes or groups to which entities may be assigned (cf. [4]). Roles, security classifications, security clearances, status levels etc., which are used in various access control models (see, for example, [6, 8, 22]), are particular instances of what we call categories. The categories of

interest, for what we propose, are application-specific and are determined by usage (individual, community or universal) rather than being defined by necessary and sufficient conditions. In financial applications, for example, there may be general categories like "creditworthy" or "bad debtor" that are part of the shared ontology of a community that is interested in testimonial knowledge about e-traders, for instance.

The part of $\mathcal{CSL}$ that we use to specify access control policies is centered on three theory-specific predicates, pca, $arca$ and par, which have following meanings (cf. [4]): $pca(p, c)$ iff the principal $p \in \mathcal{P}$ is assigned to the category $c \in \mathcal{C}$; $arca(a, r, c)$ iff the privilege of performing action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$ is assigned to the category $c \in \mathcal{C}$; $par(p, a, r)$ iff the principal $p \in \mathcal{P}$ is authorized to perform the action $a \in \mathcal{A}$ on resource $r \in \mathcal{R}$. The par relation is defined in terms of $arca$ and pca by the following rule: $\forall p, a, r, c(arca(a, r, c) \wedge pca(p, c) \rightarrow par(p, a, r,))$ that reads as follows: "If a principal p is assigned to a category c to which the privilege to perform action a on resource r has been assigned, then p is authorized to perform the a action on r".

In our community-based approach to testimonial knowledge, multiple testifiers may make assertions of their *propositional attitudes* [21] to a community-based repository (a database) of assertions, which is maintained by an oracle and constitutes a store of propositional attitudes as a set of triples (s_i, α, ψ) such that: s_i $(1 \leq i \leq n)$ is a source of assertions in a community of sources $\Sigma = \{s_1, \ldots, s_n\}$ of testimonial knowledge. ψ is a proposition; α is a propositional attitude that a source s_i has in relation to ψ (e.g., s_i "believes" ψ, s_i "disbelieves" ψ).

The triples (s_i, α, ψ) are used to represent a particular type of *that*-clause expressed in terms of a particular predicate, the pca predicate. In $\mathcal{CSL}$, we represent propositional attitudes by using a 3-place $assertion$ predicate. That is, $assertion(s_i, \alpha, \psi)$ is included, in a repository maintained by an oracle, to express that the source s_i has the propositional attitude α in relation to ψ, e.g., s_i "believes that" ψ is represented as $assertion(s_i, believes, \psi)$. For example, K_{Alice} may believe that K_{Bob} is a "bad debtor".

We restrict attention to the propositional attitudes "believes" and "disbelieves" in this paper. We interpret s_i believes ψ to be source s_i holding ψ to be "true"; by s_i disbelieves ψ we intend disbelieves to mean that s_i rejects the belief that ψ. In the case of believes and disbelieves, the propositional attitudes are subjective; they are used to express the opinions of a testimonial source.

Believes and disbelieves are propositional attitudes that a source may have in relation to the atomic proposition ψ or its negation, $\neg\psi$. In this paper, we assume the following (asymmetric) semantics for believes/disbelieves. If a source s_i asserts that it disbelieves ψ $(\neg\psi)$ then that does not commit s_i to asserting that it believes $\neg\psi$ (ψ). However, a source s_i that asserts that it believes ψ $(\neg\psi)$ implicitly asserts that it disbelieves $\neg\psi$ (ψ). In our approach, ψ is a proposition on a principal-category assignment, i.e., $pca(p, c)$. A source s believing that $\neg pca(p, c)$ believes that ψ should be prohibited from being assigned to the category c and so s must disbelieve that ψ can be assigned to c (otherwise, s would not be rational). Similarly, a source s that believes that $pca(p, c)$ must disbelieve $\neg pca(p, c)$. A source that does not assert what its propositional attitude is in relation to $pca(p, c)$ is said to *suspend* its judgement on the assignment of the principal p to category c (suspension is appropriate in the case where, for instance, s has no information about a principal-category assignment or s has conflicting evidence on what

the assignment should be). Suspension of judgement by source s on the attitude α in relation to proposition ψ just means, in our scheme, that there is no $assertion(s, \alpha, \psi)$ fact in an oracle's repository.

Another way of understanding our intended semantics is in terms of normative *ought* assertions. The testifier s_i disbelieving $pca(p, c)$ means that s_i asserts that the belief that the principal p is assigned to the category c ought to be rejected by the community. That is, disbelieving is the way in which testifiers express their dissenting to the beliefs of other testifiers in a community. The example that follows illustrates the different epistemic positions, expressed in terms of ought, that we allow.

Example 1. Consider the following scenario, in which there are four contributors of community testimony ($K_\alpha, K_\beta, K_\gamma$ and K_δ) on a principal K_{Alice} and one category $preferred$:

> *The contributor K_α believes that K_{Alice} ought to be assigned to the category $preferred$ because K_{Alice} satisfies K_α's criterion for that assignment. In contrast, it is K_β's position that the community ought to disbelieve the assertion that K_{Alice} has $preferred$ status (i.e., K_β recommends rejecting the assertion that K_{Alice} should be held to be categorized as $preferred$). The testifier K_γ has had several bad experiences with K_{Alice} and therefore K_γ asserts that it believes that K_{Alice} is definitely not to be assigned to the $preferred$ category. Finally, K_δ's position is that K_{Alice} ought not to be categorized as definitely not $preferred$, but K_δ does not want to assert that K_{Alice} is definitely $preferred$.*

For this policy, the following assertions are included in an oracle's repository of testimony:

$$assertion(K_\alpha, believes, pca(K_{Alice}, preferred)).$$
$$assertion(K_\beta, disbelieves, pca(K_{Alice}, preferred)).$$
$$assertion(K_\gamma, believes, \neg pca(K_{Alice}, preferred)).$$
$$assertion(K_\delta, disbelieves, \neg pca(K_{Alice}, preferred)).$$

Notice that it is up to a community to decide what propositional attitudes it wishes to admit and how those attitudes are to be understood in the community. We also note that propositional attitudes may be tensed, e.g., "believed", indexed by time, and epistemically guarded (e.g., weakly believed or strongly disbelieved), but we do not consider such possibilities in this paper because of limitations on space.

3 Distributed Architecture for Policy Specification

In Figure 1, the architecture of our abstract access control framework is illustrated in overview. There are four main types of entities: *resources, acceptors, oracles* and *contributors*. As we have said, assertions of propositional attitudes on pca definitions are contributed by community members to oracles and are then used by acceptors, of an oracle's assertions, to define authorizations in terms of par. Acceptors act as reference monitors and define the policy to access a guarded resource. Oracles serve a community of contributors (i.e., principals) in terms of making assertions that are of value to

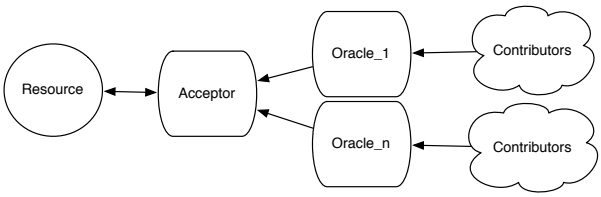

Fig. 1. Abstract Model

acceptors in evaluating access requests. Contributors or other external principals may request access to resources that are protected by acceptors As our concern is to capture a community-based view of testimony, we need to be able to talk about all members of a community having a particular propositional attitude α on proposition ψ, or some source having attitude α on ψ, or the majority attitude on ψ being α. For that, we add "counting operators" to $\mathcal{CSL}$. These operators range over the pca predicate and have a semantics that may be informally understood as follows: $\Box^{\alpha}(pca(p,c))$ is "true" iff every source of testimonial knowledge in a community Σ has the attitude α in relation to the assignment of a principal p to category c; $\Diamond^{\alpha}(pca(p,c))$ is "true" iff some source of testimonial knowledge in Σ has the attitude α in relation to $pca(p,c)$; $M^{\alpha}(pca(p,c))$ is "true" iff the majority of sources of testimonial warrant in Σ have the attitude α on $pca(p,c)$. As we restrict attention to the propositional attitudes "believes" and "disbelieves" in this paper, assertions about aggregations of pca will henceforth be expressed in the form $\bigcirc^{B^+}(pca(p,c))$ or $\bigcirc^{B^-}(pca(p,c))$ where $\bigcirc \in \{\Box, \Diamond, M\}$ and B^+ is short for "believes that" and B^- is short for "disbelieves that". In addition, we introduce into $\mathcal{CSL}$ a particular connective @ for external query evaluation over remote policy bases such that φ @ ω expresses that: "At remote source ω, φ is true", where φ can be either a literal or an aggregate function.

For the specification of the access control policies that we will introduce, we use standard ASP-DLV syntax with aggregate functions (in the sense of [10]). An *aggregate function* has the form $f(S)$ where S is a set, and f is a function name from the set of function names $\{\#count, \#sum, \#max, \#min, \#times\}$.

Definition 1 (Access Control Policy). *An access control policy is a set of rules of the form $h \leftarrow b_1, \ldots, b_n$ where, h is a literal L or a counting operator applied to an instance of $\bigcirc pca(\_,\_)$ with $\bigcirc \in \{\Box^{\alpha}, \Diamond^{\alpha}, M^{\alpha}\}$; $\alpha \in \{B^+, B^-\}$ and;*

$$b_i := (not)L \mid (\neg)\bigcirc(\neg)pca(\_,\_) \mid assertion(\_,\_,(\neg)pca(\_,\_)) \mid L \,@\, \omega \mid$$
$$L_g \prec_1 f(S) \prec_2 R_g \mid L_g \prec_1 f(S) \,@\, \omega \prec_2 R_g$$

We note that $L_g \prec_1 f(S) \prec_2 R_g$ is an *aggregate atom* where $f(S)$ is an aggregate function, $\prec_1, \prec_2 \in \{=, <, \leq, >, \geq\}$; L_g and R_g (called *left guard*, and *right guard*, respectively) are terms; and $\neg$ is strong negation [3]. We use aggregates and comparison operators on the numbers of sources of testimonial knowledge to allow acceptors to specify flexibly and in high-level terms the degrees of testimonial support that are required by them in deciding access requests (e.g., whether all, some, or a majority of testifiers have the attitude α with respect to $pca(p,c)$). As is conventional, variables in policy rules will appear in the upper case; constants are in lower case. We also restrict attention to policies with non-recursive aggregates [12] that are defined as a finite set of safe stratified clauses. This means that our access control policies have a unique answer

set. We assume that all community members express their access control policies in a language that has a unique answer set as its intended meaning.

A policy rule of the form $\bot \leftarrow b_1 \ldots b_n$ (where $\bot$ is *falsum* and denotes an arbitrary contradiction) is called a *constraint*; a constraint is to be understood (informally) as asserting that it is impossible for $b_1 \ldots b_n$ to be "true" (or provable) simultaneously.

The pca predicate occurring, within the scope of a counting operator, may be negated by using the strong negation operator $\neg$ (but not the weak negation operator not [3]). On this, we may have: $\Box^{\alpha}(\neg pca(p, c))$ is "true" for a community Σ iff every source of testimonial knowledge in Σ adopts the attitude α on principal p not being assigned to category c; $\Diamond^{\alpha}(\neg pca(p, c))$ is "true" for a community Σ iff some source of testimonial knowledge in Σ adopts the attitude α on principal p not being assigned to category c; $M^{\alpha}(\neg pca(p, c))$ is "true" for a community Σ iff the majority of sources of testimonial knowledge in Σ have the attitude α about principal p not being assigned to category c; Moreover, strong negation may be applied to the counting operators that we admit: $\neg\Box^{\alpha}(pca(p, c))$ is "true" for a community Σ iff it is not the case that every source of testimonial knowledge in Σ asserts that it has the attitude α on $pca(p, c)$; $\neg\Diamond^{\alpha}(pca(p, c))$ is "true" for a community Σ iff no source of testimonial knowledge in Σ asserts that it has the attitude α about $pca(p, c)$; $\neg M^{\alpha}(pca(p, c))$ is "true" for a community Σ iff it is not the case that the majority of sources of testimonial knowledge in Σ assert that they have the attitude α about $pca(p, c)$.

The meanings above for $\neg\Box^{\alpha}$, $\neg\Diamond^{\alpha}$ and $\neg M^{\alpha}$ extend to $\neg pca(p, c)$. For example, $\neg\Box^{\alpha}(\neg pca(p, c))$ is "true" for a community Σ iff not every source of testimonial knowledge in Σ has the attitude α on the proposition $\neg pca(p, c)$ (and *mutatis mutandis* for the other cases).

It should be noted that, for any literal L, $not\ L$ and $not\ \neg L$ are the only uses of the nonmonotonic negation-as-failure (NAF) not operator that we allow in $\mathcal{CSL}$, iterated modalities are not permitted in the form of $\mathcal{CSL}$ used in this paper, and uses of double $\neg$-negation are not permitted at all in $\mathcal{CSL}$.

By using counting functions, we are able to define the semantics of $\Box^{\alpha}$, $\Diamond^{\alpha}$, various forms of the M^{α} operator, additional operators for policy specification and a range of comparison conditions on the assertions made by contributing testifiers. We also allow arithmetic and comparison operators in the set, $\{<, \leq, >, \geq, +, -, \div, \times\}$ in $\mathcal{CSL}$. These operators, which have their usual interpretation on numbers, may also be used to express access control requirements, e.g., if the number of testifiers that believe that $pca(K_{\alpha}, c')$ holds exceeds the number that believe $\neg pca(K_{\alpha}, c')$ by at least 5 then source s' accepts that K_{α} is assigned to the category c'. Moreover, several connections between operators can be identified, e.g., $\Diamond^{\alpha} pca(P, C) \leftarrow \Box^{\alpha} pca(P, C)$ holds for a non-empty community of testifiers and additional operators may be added to more expressive forms of $\mathcal{CSL}$. For example, $W^{\alpha}\ pca(p, c)$ may be introduced to define a "minority" operator, possibly expressed in terms of M^{α}: $W^{\alpha} pca(p, c) \leftarrow \neg M^{\alpha} pca(p, c)$.

As we have said, a community decides what operators, propositional attitudes, and literals it wishes to talk in terms of when expressing access control requirements and therefore what general rules (axioms) apply to the community of users. The following set of rules apply to the interpretation of community security that we have described (where the notation $(\neg) pca(p, c)$ is to be understood as allowing either $\neg pca(p, c)$ or $pca(p, c)$ literals to be expressed, but not both):

(1) $\bot \leftarrow L \wedge \neg L$
(2) $\bot \leftarrow \neg \bigcirc (\neg) pca(P, C) \wedge \bigcirc (\neg) pca(P, C)$
(3) $\neg \Diamond^{\alpha} \neg pca(P, C) \leftarrow \Box^{\alpha} pca(P, C)$
(4) $\neg \Box^{\alpha} \neg pca(P, C) \leftarrow \Diamond^{\alpha} pca(P, C)$
(5) $\bot \leftarrow assertion(S, believes, pca(P, C)), assertion(S, disbelieves, pca(P, C))$
(6) $assertion(S, disbelieves, \neg pca(P, C)) \leftarrow assertion(S, believes, pca(P, C))$
(7) $assertion(S, disbelieves, pca(P, C)) \leftarrow assertion(S, believes, \neg pca(P, C))$
(8) $\neg \Diamond^{B^-} ((\neg) pca(P, C)) \leftarrow \Box^{B^+} ((\neg) pca(P, C))$
(9) $\Box^{B^-} (pca(P, C)) \leftarrow \Box^{B^+} (\neg pca(P, C))$
(10) $\neg \Diamond^{B^+} (pca(P, C)) \leftarrow \Box^{B^-} (pca(P, C))$

Constraint (1) is standard in ASP languages and forces truth assignments to literals to be consistent. Rule (2) is similar to (1) but it applies to any of the operators that we admit (i.e., $\bigcirc \in \{\Box^{\alpha}, \Diamond^{\alpha}, M^{\alpha}\}$). Rules (3) and (4) model the duality of the $\Box$ and $\Diamond$ operators. Constraint (5) says that no source in a community of testifiers can have the propositional attitude of both believing and disbelieving $pca(P, C)$. Rules (6) and (7) capture the semantics of the consequences of believing principal-category assignments, which we described above. Rules (8), (9) and (10) describe the duality of $\Box$ and $\Diamond$ for the counting operators.

4 Oracle Policy Specification

In this section, we illustrate, by examples, the use of oracle policy assertions and the different representations of pca that oracles may specify. The example that follows illustrates the former.

Example 2. Consider the following fragment of policy information $\mathcal{I}_1$ that needs to be represented by an oracle, denoted by ω:

> *Every contributor source of testimonial knowledge, to ω, believes that the principal K_{α} is assigned to the category $c_1 \in \mathcal{C}$, there is at least one contributor source of testimonial knowledge on assertions of principal-category assignments that believes that the principal K_{β} is assigned to $c_2 \in \mathcal{C}$, the majority of contributor sources of testimonial knowledge on assertions of principal-category assignments believe that the principal K_{γ} is assigned to $c_3 \in \mathcal{C}$, every contributor source of testimonial knowledge believes that the principal K_{δ} is not assigned to $c_4 \in \mathcal{C}$, and no contributor source says that the principal K_{ϵ} is assigned to the category $c_5 \in \mathcal{C}$.*

To represent the information in $\mathcal{I}_1$, the oracle ω may include the following assertions on aggregated testimony:

$$\Box^{B^+} (pca(K_{\alpha}, c_1)). \quad \Diamond^{B^+} (pca(K_{\beta}, c_2)). \quad M^{B^+} (\, pca(K_{\gamma}, c_3)).$$
$$\Box^{B^+} (\neg pca(K_{\delta}, c_4)). \quad \neg \Diamond^{B^+} (pca(K_{\epsilon}, c_5)).$$

In Example 2, the oracle ω expresses its community-based testimony in aggregated form. It should be clear that $\mathcal{CSL}$ is intended to be a high-level *specification* policy language for community-based testimony. This is one reason why we choose to define $\Box$, $\Diamond$ and M in terms of counting functions rather than using $count$ explicitly in policy

specifications. An atom like $\Box^{B^+}(pca(K_\alpha, c_1))$ is a simplification of a requirement to "count the number of community members that believe that K_α ought to be assigned to the category c_1 and if this count is greater than the total number of contributors then the atom is true". The need to have higher-level operators becomes more acute when a range of majority operators is admitted in $\mathcal{CSL}$.

The next example, illustrates what is possible in terms of policy representation by combining individual assertions on testimony with aggregate testimony from remote sources.

Example 3. Consider the following policy information $\mathcal{I}_2$ maintained by an oracle ω:

> *The consequent $M^{B^+}(pca(P, c_5))$ holds if the majority of sources of testimonial knowledge for the oracle ω_1 believe that P is assigned to the category $c_6 \in \mathcal{C}$, the majority of sources of testimonial knowledge for the oracle ω_2 disbelieve that P is assigned to $c_7 \in \mathcal{C}$, and it fails to be the case that ω's specific contributor source s' believes $\neg pca(P, c_5)$.*

To represent the information in $\mathcal{I}_2$, ω may use the following access control policy rule:

$$M^{B^+}(pca(P, c_5)) \leftarrow M^{B^+}(pca(P, c_6)) \mathbin{@} \omega_1, M^{B^-}(pca(P, c_7)) \mathbin{@} \omega_2, \\ not(assertion(s', B^+, \neg pca(P, c_5))).$$

A feature of our approach is that different definitions of the $\Box$, $\Diamond$ and M operators can be naturally accommodated. For example, the careful reader will have noted that, in what we have said thus far, implicitly an oracle maintains a unary relation $source$ (say), the extension of which is the domain of identifiers of sources from which the oracle gathers its testimonial knowledge. In this case, "every source" has the attitude α in relation to $pca(p, c)$ implies quantification over the elements in the extension of $source$. Similarly, $\Diamond^\alpha pca(p, c)$ corresponds to existential quantification over the elements in the extension of $source$. However, as intimated, these are not the only possibilities. Instead of $source/1$, for example, an oracle may maintain a binary form, $source(s, c)$, with the following semantics: *s is a source of testimonial knowledge on the category c.* In this case, quantification is possible with respect to sources that make assertions about *particular categories*. The difference between $source(s)$ and $source(s, c)$ is needed if it is the case that not all sources that contribute to the oracle's "knowledge" necessary make assertions about the category c. In the case where $source(s, c)$ is relevant, the semantics of $\Box$ changes to the following:

> *$\Box^\alpha_c(pca(p, c))$ is "true" according to an oracle ω iff every source that declares to ω that it is a source of testimonial knowledge on c has the attitude α in relation to $pca(p, c)$.*

The next example, illustrates the potential use of this additional operator.

Example 4. An oracle ω may express the following access control policy information using $\mathcal{CSL}$:

$$assertion(\omega, B^+, pca(P, C)) \leftarrow source(S, C), not(assertion(S, B^+, \neg pca(P, C)))$$
$$assertion(\omega, B^+, \neg pca(P, C)) \leftarrow source(S, C), not(assertion(S, B^+, pca(P, C)))$$

As it is defined, $\neg pca(P, C)$ holds according to ω if some source S of testimonial knowledge on the assignment of a principal P to the category C does not believe that

the principal P is assigned to C or if some source of testimonial knowledge on C does not believe that $\neg pca(P, C)$ holds.[2]

The M^{α} operator may also be defined in multiple ways, e.g., the oracle ω may specify,

$$M^{B^+}(pca(P, c_0)) \leftarrow (\#count\{S : assertion(S, B^+, pca(P, c_1)\} \ @ \ \omega_1) > (\#count\{S : assertion(S, B^+, pca(P, c_2))\} \ @ \ \omega_2)$$

to express that the majority of the community contributors of testimonial knowledge believe $pca(P, c_0)$ if the number of contributors to the oracle ω_1 that believe $pca(P, c_1)$ is greater than the number of contributors to the oracle ω_2 that believe $pca(P, c_2)$. Perhaps, for example, ω assigns a principal P to the category c_0 (the $potential\_customer$ category, say) if the number of testifiers that contribute opinions to the oracle ω_1 believe that P is assigned to the category c_1 (the $preferred\_customer$ category, say) and their collective opinion outweighs the number of sources of testimony, that contribute to ω_2, that believe that P is assigned to the category c_2 (a $blacklisted$ category, say).

5 Acceptor Policy Specification

Having described oracle-based access control policy specification in the previous section, we now describe local policy formulation by acceptor agents that make use of an oracle's assertions. In $\mathcal{CSL}$, an acceptor agent defines the par relation (i.e., authorizations) thus: $par(P, A, R) \leftarrow pca(P, C), arca(A, R, C)$.

That is, the principal P has the access privilege A (or P can perform the action A) on the resource R if P is assigned to a category C to which the permission (A, R) is assigned. For example, a principal categorized as "sales manager" is authorized to read the "monthly sales file" msf if the permission $(read, msf)$ is assigned to members of the "sales manager" category. The categories that an acceptor chooses to use to define authorizations that apply locally are defined by it in terms of the categories used to classify principals within a community of sources of testimonial knowledge.

Example 5. Consider the following local access control policy requirements of an acceptor of assertions on principal-category assignments, which are defined with respect to a remotely located oracle ω:

> *For a principal P to be assigned to the acceptor's local category c_0, all contributors to the oracle ω must believe that $pca(P, c_1)$ and at least one member of the community must believe that P is assigned to the category c_2 (according to ω). Moreover, the specific contributor s_0 must assert (via ω) that P is assigned to the category c_3 and the specific contributor s_1 must assert (via ω) that she disbelieves that P should not be assigned to the category c_4.*

To represent these access control requirements, the following policy specification may be defined by an acceptor:

$$pca(P, c_0) \leftarrow \Box^{B+}(pca(P, c_1)) \ @ \ \omega, \Diamond^{B+}(pca(P, c_2)) \ @ \ \omega.$$
$$assertion(s_0, B^+, pca(P, c_3)) \ @ \ \omega, assertion(s_1, B^-, \neg pca(P, c_4,)) \ @ \ \omega.$$

[2] Note that the repository of assertions that is maintained by an oracle may contain assertions made by any source in a community including an oracle.

Although we take each $arca$ definition to be local to the acceptor site that controls access to the resource, a community-based approach to $arca$ specifications is possible. It is also possible for acceptors to specify $\neg arca$ definitions, to express that a permission is not to be assigned to a category or the permission is denied. When $\neg arca$ definitions are used, variants of par are possible (e.g., to represent "open" or "closed" policies with different conflict resolution strategies). For example,

$$par(P, A, R) \leftarrow pca(P, C), not\ \neg arca(A, R, C).$$

may be used to express that a principal P has the A access privilege on resource R if P is assigned to a category C and the (A, R) permission is not denied to C.

Moreover, par definitions may be specialized depending on the particular assertions stored by oracles. The example that follows next illustrates this.

Example 6. Consider the following acceptor site's specification of two forms of authorizations:

$$par_1(P, A, r_1) \leftarrow M^{B+}(pca(P, C))\ @\ \omega, arca(A, r_1, C).$$
$$par_2(P, A, r_2) \leftarrow M^{B+}(pca(P, c'))\ @\ \omega, \neg\Diamond^{B^-}(pca(P, c'))@\ \omega, arca(A, r_2, c').$$

The definition of par_1 may be read as follows: "any principal P is permitted to perform any action A on the resource r_1 if the majority of contributors to oracle ω believe that $pca(P, C)$ holds and $arca(A, r_1, C)$ holds locally". The definition of par_2 says: "Any principal P is permitted to perform any action A on the resource r_2 if the majority of contributors to ω believe that $pca(P, c')$ (i.e., the principal is assigned to category c'), none of ω's contributors disbelieve $pca(P, c')$ at ω, and $arca(A, r_2, c')$ holds locally.

The sceptical reader may argue that our approach violates the privacy of principals whose category assignments are shared in an oracle. Nevertheless, we envisage principals being able to opt-out of this type of information sharing if they wish. The sceptic may argue that our use of the nonmonotonic not operator is "unsafe" because authorizations may hold by default, in the absence of knowledge. However, negation-as-failure is used iff it is appropriate for specific policies and for specific definitions of authorizations, e.g., where access is allowed in the absence of a disbelief holding. The sceptic may suggest that what we propose bears no relation to "real" access control models or policies. However, policies defined in terms of RBAC, MAC, DAC, Attribute-based access control, status-based access control etc. can all be represented within our category-based framework (cf. [4]). That is, what we describe is an approach that extends access control models to the community-based level that we are introducing. The sceptic may argue that $\mathcal{CSL}$ is merely an abstract specification language for expressing distributed access control policies and that $\mathcal{CSL}$ has an equivalent representation in existing policy specification languages. However, as we will show next, policies that are specified in $\mathcal{CSL}$ translate into (various) practical languages for implementation and in the related work section we explain why $\mathcal{CSL}$ differs to existing proposals of policy specification languages.

6 Practical Issues

In this section, we present secommunity,[3] an implementation of our framework in the DLV system and some performance measures for it. We then consider proving properties of policies in our scheme. In what follows, DLV code fragments are presented using monospace font. Henceforth, we view acceptors, oracles and contributors as DLV knowledge bases (KBs). Counting operators can be implemented by using DLV counting functions. For instance, $\Box^{B^+}$, $\Diamond^{B^+}$ and M^{B^+} can be defined as follows[4]:

$$\begin{aligned}
&\texttt{box(believes, pca(P, C)) :- assertion(_, believes, pca(P, C)),}\\
&\texttt{\#count\{S : assertion(S, believes, pca(P, C)), source(S)\} = R,}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\texttt{\#count\{Y : source(Y)\} = R.}\\
&\texttt{diamond(pca(P, C)) :- assertion(S, believes, pca(P, C)), source(S).}\\
&\texttt{majority(pca(P, C)) :- assertion(_, believes, pca(P, C)),}\\
&\texttt{\#count\{S : assertion(S, believes, pca(P, C)), source(S)\} = R1,}\\
&\qquad\qquad\qquad\texttt{\#count\{S : source(S)\} = R2, R3 = R2/2, R1 > R3.}
\end{aligned}$$

To represent the distributed aspect of our approach, we defined in DLV-Complex[5] two external predicates (also called built-ins):

- $\#\texttt{at}(\texttt{IP}, \texttt{G})$ which reads as: "At the remote DLV source indexed by IP, literal G holds."
- $\#\texttt{f\_at}(\texttt{IP}, \{\texttt{P} : \texttt{Conj}\}, \texttt{PX})$, which has several readings depending on the aggregate function f. For instance, If $\texttt{f} = \texttt{count}$, then PX is unified with the value V, the sum of the elements in $\{\texttt{P} : \texttt{Conj}\}$ at the remote DLV source indexed by IP, i.e., $\#\texttt{count}\{\texttt{P} : \texttt{Conj}\} = \texttt{V}$.[6]

For each external predicate (e.g., #at, #count_at) we associated a C++ program that queries the remote DLV knowledge base indexed by IP. For instance, the rule

$\texttt{L} \text{ :- } \#\texttt{count\_at}(\text{"IP"}, \text{"}\texttt{S} : \texttt{assertion}(\texttt{S}, \texttt{believes}, \texttt{pca}(\texttt{p}, \texttt{c}))\text{"}, \texttt{PX}), \texttt{PX} > 3$

reads as follows: "If the remote KB located at address IP contains at least three sources that believe that p is assigned to category c, then literal L holds true". Due to space constraints, we do not discuss the implementation details.[7] In order to assess the overhead caused by the evaluation of external predicates, we studied the relationship between execution time and the number of external predicates necessary to evaluate a given rule. In particular, we consider the evaluation of rules of the type

$$\begin{aligned}
(T_n)\quad &\texttt{L} \text{ :- } \#\texttt{count}(\texttt{IP}, \texttt{S} : \texttt{assertion}(\texttt{S}, \texttt{believes}, \texttt{pca}(\texttt{p}_1, \texttt{c}_1)), \texttt{PX}), \ldots,\\
&\#\texttt{count}(\texttt{IP}, \texttt{S} : \texttt{assertion}(\texttt{S}, \texttt{believes}, \texttt{pca}(\texttt{p}_\texttt{n}, \texttt{c}_\texttt{n})), \texttt{PX}), \texttt{PX} > \texttt{m}
\end{aligned}$$

Informally, the literal L holds by rule (T_n) if there are at least m sources at a remote KB (located at IP) that believe $\texttt{pca}(\texttt{p}_1, \texttt{c}_1)$, $\texttt{pca}(\texttt{p}_2, \texttt{c}_2)$, ..., $\texttt{pca}(\texttt{p}_\texttt{n}, \texttt{c}_\texttt{n})$. In Figure 2, we report the main results of our experiments.[8] For every instance of rule $T_{i\cdot 5}$ (with

[3] A preliminary version of secommunity with a simplified language has been presented as a short paper in [5].

[4] It may be that counting operators depend on other remote knowledge bases as in Example 3.

[5] https://www.mat.unical.it/dlv-complex

[6] Similarly for $\texttt{f} \in \{\texttt{sum}, \texttt{times}, \texttt{min}, \texttt{max}\}$

[7] A working implementation with source code, documentation and examples is available at http://www.di.unito.it/~genovese/tools/secommunity/index.html

[8] We conduct our tests on the WAN of University of Torino, the client is a Mac Book Pro with a Intel Core 2 Duo, 4 GB of RAM running Mac OS X 10.6.5. As a server we used a Linux workstation with AMD Athlon, 2 GB of RAM running Ubuntu 10.4.

$1 \leq i \leq 20$), we plot five points representing the average and the standard deviation of the amount of time (in milliseconds) needed to evaluate 30 runs of $T_{i \cdot 5}$ on five different KBs of increasing numbers of belief assertions. In the first KB, there are 100 sources that believe $\texttt{pca}(\texttt{p}_1, \texttt{c}_1), \ldots, \texttt{pca}(\texttt{p}_n, \texttt{c}_n)$, in the second the sources supporting `pca` are 200, and so on up to 500. The results reported in Figure 2 offer some evidence to suggest that our approach is scalable for realistic applications. It is important to note that the time to compute in DLV-Complex T_n grows *linearly* in n and that the time to compute a given rule T_i grows linearly in the size of the KB. We note too that it takes about four seconds to query 100 times a remote KB of 500 assertions with the external predicate `#count`.

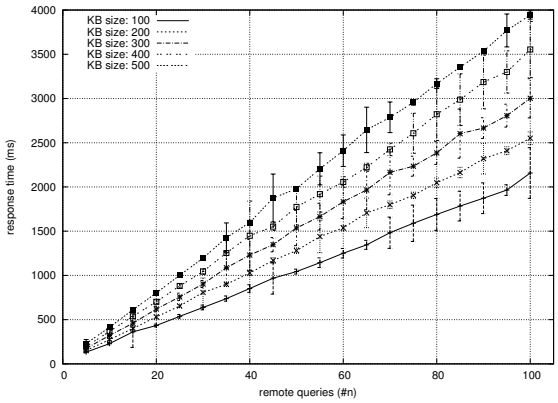

Fig. 2. Scalability of External Predicates

An important practical aspect of our approach is that $\mathcal{CSL}$ admits the possibility of proving properties of policies. To support this assertion, we briefly describe two (of several) types of policy-proof analysis in our framework: *authorization* and *rule-based policy analysis*.

Definition 2 (Access Control Problems). *Consider a policy represented by a set of $\mathcal{CSL}$ rules Π and a set of facts D then,*

- *The* authorization problem *consists in determining whether a principal p_0 is authorized to access resource r_0, which necessitates computing the answer set of $\Pi \cup D$ and checking whether $par(p_0, a_0, r_0)$ is contained in it.*
- *The* rule-based policy analysis *problem consists in taking as input the answer set of a policy and checking whether it is closed under a policy property that is expressible as a rule in $\mathcal{CSL}$.*

Theorem 1 (Complexity of Access Control Problems). *Given a policy $\Pi \cup D$, the problems reported in Definition 2 can be solved in time* polynomial *in the size of D.*

Proof. (Sketch) Both problems are data complexity *problems [2] for which it is known that the complexity of stratified ASP logic programs is polynomial. Moreover, from [12], the types of aggregate functions admitted in $\mathcal{CSL}$ do not increase the complexity of non-disjunctive ASP programs.*

Suppose next that an acceptor's policy $\Pi \cup D$ is described by a single file "`policy.txt`", which contains the policy expressed as a list of `secommunity` rules and a set of facts. The authorization problem can be solved by letting DLV compute the answer set α of $\Pi \cup D$ by running it on `policy.txt` and then parsing the DLV output looking for the required occurrence of par in α. On the rule-based policy analysis, suppose that a policy administrator wishes to check whether the answer set of `policy.txt` is closed under a given $\mathcal{CSL}$ rule $H \leftarrow L_1 \wedge \ldots \wedge L_n$. Then, it is sufficient to run DLV on `policy.txt` extended with constraint $:\!-\mathtt{L_1}, \ldots, \mathtt{L_n}, \mathtt{not\ H}$[9] and to check whether the resulting answer set is empty (i.e., the policy does not satisfy the property) or not (i.e., the policy satisfies the property). Suppose next that a policy administrator wanted to check on `policy.txt` that for any principal p, if p is authorized on action a_1 over resource r_1 then, p is authorized also on action a_2 over resource r_2. For that, it is sufficient to run DLV on `policy.txt` by adding $:\!-\mathtt{par(P, a_1, r_1)}, \mathtt{not\ par(P, a_2, r_2)}$ as a constraint. Additionally, suppose that a check of a static separation of duty property of the following type is required: "no principal can be granted for both action a_1 and action a_2 over the same resource r_1". For that, it is sufficient to run DLV on `policy.txt` by adding the following constraint $:\!-\mathtt{par(P, a_1, r_1)}, \mathtt{par(P, a_2, r_1)}$.

As a final remark, we emphasize that policy properties like safety and availability [7] follow from the soundness and completeness results of the DLV framework w.r.t. stable model semantics.

7 Related Work

We have described an extension to the work from [4] that allows for multiple distributed access control policies to be represented in a common framework and that allows for different propositional attitudes to be expressed by agents in relation to propositions of relevance to access control. The work described in [4] is concerned neither with distributed access control nor with propositional attitudes.

On the literature on logic programming for access control, our work is related to Jajodia et al.'s specification of the flexible authorization framework in logic programming terms [14] and to Barker and Stuckey's CLP-based approach [7] for access control policy specification. However, both of these approaches assume a centralized access control system and neither of them can express the range of policies that can be expressed in terms of the meta-model that we described in [4]. Neither approach allows for explicit negation to be used in policy specification, a community-based view of testimonial knowledge for authorization, the use of propositional attitudes for expressing policy information, or the use of a range of counting operators for policy specification.

The RT [18] family of trust management languages is also related to our proposal. However, RT is Datalog-based and the various RT languages do not admit the possibility of sources issuing negative assertions and they do not admit a default form of negation. RT is based on a monotonic semantics. RT also does not take account of the idea of a community-based source of knowledge (role names could refer to oracles but a doxastic element would need to be added to RT to capture what we propose) or different strengths of knowledge expressed in terms of community-based assertions.

[9] Notice that $L_1, \ldots, L_2$ may also be remote querying operators or aggregate functions.

SD3 [15] and Binder [11] are also monotonic, Datalog-based languages for expressing policies for distributed access control. As such, neither of these approaches allows for nonmonotonic access controls that we have considered in this paper. The work by Wang and Zhang [23] uses ASP, as we do, for distributed access control policy specification, but their work is based on the use of ASP to implement a variant of D2LP [18]. Our focus is quite different. We also note that RT^T allows for threshold structures that require principals from a single set to agree on an entity having a role's attribute. In $\mathcal{CSL}$, policy requirements that require all contributors to an oracle must agree that a principal-category assignment holds, but also disbeliefs, degrees of disagreement between believers and disbelievers, etc. can be expressed and so too can negated forms of belief and disbelief.

There are several additional logic-based languages (e.g., ABLP logic, DKAL and SecPAL) that, like Binder, attach special importance to a **says** construct (or a variant of it). Put simply, A **says** p is used to associate a principal with an utterance p. At a superficial level, the reader might suggest that A **says** p means A "believes" p (though that this doxastic interpretation is to be adopted it is far from being a universal view and SecPAL, in particular, assumes that principals assert "facts" not beliefs) and that a says-logic enables the concepts expressed in $\mathcal{CSL}$ to be equivalently represented. However, none of ABLP logic, DKAL or SecPAL consider disbeliefs or the subtleties of the interactions between beliefs and disbeliefs. In contrast, we give two distinctions between beliefs and disbeliefs and we give axioms to make plain the formal relationships between them. We also make clear how we interpret the notion of suspension (of beliefs). None of ABLP logic, DKAL or SecPAL are concerned with the notion of trust being grounded in a community view and being represented by using assertions expressed using aggregates. Moreover, all reasoning methods for these logics are *local*, i.e., they do not offer primitives to query distributed knowledge bases. A notable exception is Soutei [20], which is an extension of Binder that accepts remote fetching of assertions. However, Soutei is monotonic, it is based on **says** modality, it does not support aggregate functions and it is not clear that it offers a client-server architecture that can be deployed in real-world scenarios (e.g., the Internet).

Our work is also related to efforts, like SPKI/SDSI [9], that are concerned with certified authorization in distributed computing contexts. On this, Howell and Kotz [13] discuss "beliefs" in the context of their formalization of a variant of SPKI/SDSI, but our reading of their notion of belief suggests that what they have in mind is quite different to our interpretation of "believes that". Howell and Kotz give no formal definition of what they take "belief" to mean in their formalization of distributed authorization; they appear to assume beliefs to be synonymous with assertions of propositions, rather than being an indicator of an epistemic attitude to a proposition. The work by Liau et al. on BIT logic [19] is related to ours in the sense that beliefs in propositions may be expressed in a logic for representing policies for trust management. However, the social aspect of testimony is not considered by Liau et al.

There have been many approaches to trust management that are based on measures of trust (or disbelief or distrust) that are (often) expressed by using real numbers in the interval $[-1, 1]$ as measures of belief/disbelief in some proposition. However, it is often far from clear how, for example, the assignment of trust measures to propositions can be precisely and uncontroversially allocated and it is not clear how changes in

evidence justify changes in measures of belief/disbelief. In contrast, we argue that the three epistemic positions that we allow (belief, disbelief and suspension of judgement) provides a basis for defining a meaningful semantics for community-based, distributed access control policies.

8 Conclusions and Further Work

We argued that for some (not all) applications, socially constructed testimony may be usefully employed for distributed access control (that relying on individual, authoritative sources of testimony is not always appropriate). On this, we have described an approach for socially constructed trust.

We have taken testimonial knowledge as being expressed in the form of a testifier's propositional attitude in relation to propositions about principal-category assignments and we view trust in terms of community beliefs. For access control policy specification, we introduced $\mathcal{CSL}$ (Section 3). By adopting a weak negation operator, nonmonotonic policies can be represented in our framework. Acceptors and oracles express access control policy information by using the same specification language. Scalability issues are addressed by subdividing the task of providing testimony between contributors to a community view. For the shared conceptual framework, we extend the meta-model for access control as defined in [4]. To illustrate our approach, we gave a number of examples of oracle-based access control representation (Section 4) and acceptor-based authorization policies (Section 5). We described an ASP-based implementation of our approach (Section 6) and we presented some performance measures for this implementation of our approach that offer evidence of its scalability (Section 6). We argued (in Section 7) that our approach offers something different to existing approaches that either do not consider the community aspect of testimonial knowledge or that do recognize the importance of community constructed knowledge but express aggregated assertions in terms of measures of "reputation".

In future work, we propose to investigate community membership issues (e.g., how to address the effects of changes to the community). Thus far, we have only considered a "democratic" form of community in which every contributor's assertions have the same weight. Accommodating variable measures of authority in terms of the assertions that contributors make to the community view is another issue that requires further investigation. We also intend to investigate issues relating to temporally constrained propositional attitude reports and we propose to include additional types of propositional attitudes (e.g., "knows that" p by proof of p) in extended forms of our framework.

Acknowledgements. The authors would like to thank Daniele Rispoli for the implementation of the external predicates of `secommunity`.

References

1. M. Abadi. Access control in a core calculus of dependency. *Electr. Notes Theor. Comput. Sci.*, 172:5–31, 2007.
2. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

3. C. Baral and M. Gelfond. Logic programming and knowledge representation. *J. Log. Program.*, 19/20:73–148, 1994.
4. S. Barker. The next 700 access control models or a unifying meta-model? In *Procs. of SACMAT*, pages 187–196, 2009.
5. S. Barker and V. Genovese. Secommunity: A framework for distributed access control. In *Procs. of LPNMR*, pages 297–303, 2011.
6. S. Barker, M. J. Sergot, and D. Wijesekera. Status-based access control. *ACM Trans. Inf. Syst. Secur.*, 12(1), 2008.
7. S. Barker and P. Stuckey. Flexible access control policy specification with constraint logic programming. *ACM Trans. Inf. Syst. Secur.*, 6(4):501–546, 2003.
8. D. E. Bell and L. J. LaPadula. Secure computer system: Unified exposition and multics interpretation. *MITRE-2997*, 1976.
9. D. E. Clarke, J.-E. Elien, C. M. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in SPKI/SDSI. *J. Computer Security*, 9(4):285–322, 2001.
10. T. Dell'Armi, W. Faber, G. Ielpa, N. Leone, and G. Pfeifer. Aggregate functions in disjunctive logic programming: Semantics, complexity, and implementation in DLV. In *Procs. of IJCAI*, pages 847–852, 2003.
11. J. DeTreville. Binder, a logic-based security language. In *Proc. IEEE Symposium on Security and Privacy*, pages 105–113, 2002.
12. W. Faber and N. Leone. On the complexity of answer set programming with aggregates. In *Procs. of LPNMR*, pages 97–109, 2007.
13. J. Howell and D. Kotz. A formal semantics for SPKI. In *Procs. of ESORICS*, pages 140–158, 2000.
14. S. Jajodia, P. Samarati, M. Sapino, and V. Subrahmaninan. Flexible support for multiple access control policies. *ACM TODS*, 26(2):214–260, 2001.
15. T. Jim. SD3: A trust management system with certified evaluation. In *IEEE Symp. Security and Privacy*, pages 106–115, 2001.
16. N. Leone and W. Faber. The DLV project: A tour from theory and research to applications and market. In *Procs ICLP*, pages 53–68, 2008.
17. N. Li, B. N. Grosof, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. Inf. Syst. Secur.*, 6(1):128–171, 2003.
18. N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Procs. of IEEE Symposium on Security and Privacy*, pages 114–130, 2002.
19. C.-J. Liau. Belief, information acquisition, and trust in multi-agent systems–a modal logic formulation. *Artif. Intell.*, 149(1):31–60, 2003.
20. A. Pimlott and O. Kiselyov. Soutei, a logic-based trust-management system. In *Procs. of FLOPS*, pages 130–145, 2006.
21. B. Russell. On denoting. *Mind*, 149(1):479–493, 1905.
22. R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.
23. S. Wang and Y. Zhang. Handling distributed authorization with delegation through answer set programming. *Int. J. Inf. Sec.*, 6(1):27–46, 2007.