Relationship-Based Access Control Policies and Their Policy Languages

UC CPSC Technical Report 2011-990-02

Philip W. L. Fong Ida Siahaan
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada
{pwlfong, isrsiaha}@ucalgary.ca

ABSTRACT

The Relationship-Based Access Control (ReBAC) model was recently proposed as a general-purpose access control model. It supports the natural expression of parameterized roles, the composition of policies, and the delegation of trust. Fong proposed a policy language that is based on Modal Logic for expressing and composing ReBAC policies. A natural question is whether such a language is representationally complete, that is, whether the language is capable of expressing all ReBAC policies that one is interested in expressing.

In this work, we argue that the extensive use of what we call Relational Policies is what distinguishes ReBAC from traditional access control models. We show that Fong's policy language is representationally incomplete in that certain previously studied Relational Policies are not expressible in the language. We introduce two extensions to the policy language of Fong, and prove that the extended policy language is representationally complete with respect to a well-defined subclass of Relational Policies.

Categories and Subject Descriptors

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

General Terms

Security, Language, Theory

Keywords

Access control policies, modal logic, policy languages, relationship-based access control, social networks

1. INTRODUCTION

The advent of social computing introduces the world to a new paradigm of access control, in which interpersonal relationships are explicitly tracked by the protection system for the purpose of authorization. Gates coined the term ***Relationship-Based Access Control (ReBAC)*** to refer to this paradigm of access control [19]. In a typical ReBAC system, the protection state consists of a knowledge base of primitive relationships (e.g., friend) between individual users [7], and an access control policy (e.g., friend-of-friend) is expressed in terms of composite relationships induced by the primitive relationships (e.g., friend $\circ$ friend).

While previously proposed ReBAC systems are designed mostly for social computing applications [25, 8, 7, 37, 38, 18], a formal ReBAC model was proposed recently as a general-purpose access control model [16]. The model features poly-relational social networks with asymmetric relations, as well as a context-dependent authorization procedure that limits the scope of relationships to their applicable contexts. Fong articulates three benefits of the ReBAC paradigm in general, and of his ReBAC model in particular. First, formulating policies in terms of binary relationships between users supports the natural expression of authorization decisions that are based on the relative attributes of the accessor as perceived by the resource owner (e.g., trust, professional association, etc). Traditional access control systems base their authorization decisions on some unary predicates of the users (e.g., identities, roles, etc) that is defined by a central authority. The limitations of such an approach becomes evident as notions such as ***parameterized roles*** [32] (or ***role templates*** [21]) creep into RBAC systems: e.g., the parameterized role manager(john) signifies the role assumed by the manager of John. In many emerging application domains, such as social computing and Electronic Health Records Systems, in which the protection of user-contributed contents is prominent, accessibility depends not on the intrinsic attributes of the accessor, but on the relationship between the accessor and the owner (e.g., the professional relationship). ReBAC provides natural support for such situations. Second, relationship-based access control policies support a richer form of policy composition. With policies that are based on unary predicates, the only interesting form of policy composition would be boolean combinations (i.e., conjunction, disjunction and negation), which can be captured readily by, say, a role hierarchy. Binary relations, however, supports a much richer form of composition, including relational composition (friend $\circ$ friend) and transitive closure (friend$^+$). Third, a composite binary relation supports delegation of trust in a natural way: e.g., by adopting the policy friend-of-friend, I am delegating to my friends to decide who can access.

Fong also proposed a policy language for expressing ReBAC policies [16]. The language is based on a basic modal logic, and it has been shown to be capable of expressing complex, composite relationships between an owner and an accessor, such as those found in an Electronic Health Records (EHR) system [4]. One natural question to ask is whether the policy language of Fong is representationally complete, that is, whether it is capable of expressing all the ReBAC policies we would like to express. In this work, we give a negative answer to the question, we extend the policy language to cover its shortcomings, and we formally characterize the expressive power of the extended policy language. In this last respect, this research has close affinity to enforceability research, as exemplified in [36, 29, 14, 23, 39, 30, 31], which formally characterize the structural relationships between the family of policies enforceable by a given enforcement mechanism and other naturally occurring policy families.

Our contributions are threefold:

1. A distinguished feature of ReBAC is its extensive use of what has come to be known as ***relational policies*** [1]. We demonstrate, in Section 2, that a number of relational policies previously studied in the context of Facebook-style Social Network Systems [18] cannot be expressed in the policy language of Fong.
2. We introduce, in Section 3, two new features into the ReBAC policy language, and demonstrate that the resulting language is capable of expressing the abovementioned relational policies.
3. In Section 4, we formally characterize the expressiveness of the extended policy language by proving both an "upper bound" and a "lower bound" for the family of ReBAC policies expressible in the language. The upper-bound result shows that, every policy expressible in the extended policy language belongs to the family of owner-checkable policies. The lower-bound result identifies a natural subclass of relational policies, namely, finitary relational policies, that are provably expressible in the extended language: the extended language is representationally complete with respect to finitary relational policies.

2. INADEQUACY OF $\mathcal{B}$

This section provides an overview of the background on which this work is based, and formally articulate the representational incompleteness of the basic policy language of Fong [16].

2.1 Notation

Given a binary relation $R \subseteq X \times Y$ and individuals $x \in X$ and $y \in Y$, we write $R(x, y)$ iff $(x, y) \in R$. We write 2^X for the powerset of X (i.e., the set of *all* subsets of X), $[X]^k$ for the set of all subsets of X that have a cardinality k, and $[X]^{<\omega}$ for the set of all *finite* subsets of X. We write $f : X \rightharpoonup Y$ whenever f is a function with a *subset* of X as its domain and Y as its co-domain. We write $\emptyset$ for a function with an empty domain. Consider a function $f : X \rightharpoonup Y$ and individuals $x_0 \in X$ and $y_0 \in Y$. We write $f[x_0 \mapsto y_0]$ to denote the function $f' : X \rightharpoonup Y$ defined as follows: $f'(x) = y_0$ if $x = x_0$, but $f'(x) = f(x)$ if $x \neq x_0$.

2.2 Social Networks and Relation Identifiers

We assume that an SNS defines a finite set $\mathcal{I}$ of ***relation identifiers***. Each identifier denotes a type of relationships that is tracked by the system (e.g., parent-child, patient-physician, etc). A typical member of $\mathcal{I}$ is denoted by i.

A social network is essentially a directed graph with multiple kinds of edges. While individuals are represented by vertices, each kind of directed edges represents a distinct type of relationship between users. Formally, a ***social network*** G is a relational structure [5] of the form $\langle V, \{R_i\}_{i \in \mathcal{I}} \rangle$, where:

- V is a finite set of vertices, each representing an individual in the social network.
- $\{R_i\}_{i \in \mathcal{I}}$ is a family of binary relations. The binary relation $R_i \subseteq V \times V$ specifies the pairs of individuals participating in relationship type i.

We write $V(G)$ and $R_i(G)$ respectively for the V and R_i components of the social network G.

Standard graph-theoretic concepts apply to social networks in an expected manner. We outline a few to fix thoughts. We write $G \subseteq G'$ whenever $V(G) \subseteq V(G')$ and $R_i(G) \subseteq R_i(G')$ for every $i \in \mathcal{I}$. G is said to be a ***subgraph*** of G'. Suppose $U \subseteq V(G)$. We write $G[U]$ to denote the social network $\langle U, \{R'_i\}_{i \in \mathcal{I}} \rangle$, where $R'_i = R_i(G) \cap (U \times U)$. $G[U]$ is said to be the ***subgraph of G induced by U***. Two social networks G, G' are ***isomorphic*** iff there is a bijective function $\pi : V(G) \to V(G')$ such that $(u, v) \in R_i(G)$ whenever $(\pi(u), \pi(v)) \in R_i(G')$. In this case, we write $G \cong G'$, and call π an ***isomorphism*** between G and G'.

Given a social network G, we define the ***acquaintance graph*** $\mathsf{acq}(G)$ to be the simple graph $\langle V(G), E \rangle$, where $E = \{\{u, v\} \in [V(G)]^2 \mid \exists i \in \mathcal{I} \,.\, (u, v) \in R_i(G)\}$. That is, an acquaintance graph shares the same vertex set as the social network, and there is an undirected edge between two *distinct* vertices in the acquaintance graph iff there is a typed, directed edge between the same pair of vertices in the social network. Note that, although loops may occur in the social network, the acquaintance graph is a simple graph, and thus it contains no loops. The purpose for defining the acquaintance graph is to reuse connectivity-related concepts from standard graph theory. Specifically, notions such as distance, connectedness, and components can be applied to a social network. For example, two vertices in a social network G are said to be ***connected*** iff there is an undirected path[1] between them in $\mathsf{acq}(G)$; the ***distance*** between two connected vertices in G is the length of the shortest path between them in $\mathsf{acq}(G)$; an ***(extended) neighbourhood*** of a vertex u in G is the set $N^*_G(u)$ of vertices connected to u in $\mathsf{acq}(G)$; the ***component*** of G to which u belongs is the social network $C_G(u)$, which is

[1]We consider a vertex to be connected to itself by a length-zero path. Consequently, u is always a member of $N^*_G(u)$.

defined to be $G[N^*_G(u)]$. Lastly, we write $C_G(u; v)$ to denote the social network $G[N^*_G(u) \cup \{v\}]$, which is called an ***augmented component***. The idea is to form the component of G to which u belongs, but with the augmentation of vertex v. If u and v are connected, then $C_G(u) = C_G(u; v)$. Otherwise, $C_G(u; v)$ contains $C_G(u)$ plus an additional, isolated vertex v.

Suppose $\mathcal{U}$ is a countable set of user identifiers. We denote by $\mathcal{G}(\mathcal{U}, \mathcal{I})$ the set of all *finite* social networks defined for user set $\mathcal{U}$ and relation identifier set $\mathcal{I}$. That is, $\mathcal{G}(\mathcal{U}, \mathcal{I}) = \{\langle V, \{R_i\}_{i\in\mathcal{I}}\rangle \mid V \in [\mathcal{U}]^{<\omega}, R_i \subseteq V \times V\}$. $\mathcal{G}(\mathcal{U}, \mathcal{I})$ contains all the social networks with a vertex set that is a finite subset of $\mathcal{U}$. Since $\mathcal{I}$ is finite, such a social network has only finitely many edges. Note that, although every member of $\mathcal{G}(\mathcal{U}, \mathcal{I})$ is finite in size, the size of such a member is not bounded.

2.3 ReBAC Policies

In [16] a formal model of ReBAC is introduced. The state of a ReBAC system is a collection of social networks, as defined above. State transition involves the mutation of the social networks (e.g., addition and deletion of relationship edges). An access attempt involves an ***accessor*** requesting access to a ***resource*** owned by an ***owner***[2]. When an access is attempted at a given state, the policy that controls the accessibility of the requested resource will be evaluated against a certain social network induced by the current state. The policy either allows or denies the request. In the following, we formally specify what a ReBAC policy is.

A ReBAC policy defines a desired binary relation between resource owners and resource accessors in the context of a given social network. Suppose we are given a fixed set $\mathcal{U}$ of users and a fixed set $\mathcal{I}$ of relation identifiers. A ***ReBAC policy*** is a family of binary relations over $\mathcal{U}$, indexed by social networks from $\mathcal{G}(\mathcal{U}, \mathcal{I})$. More specifically, a policy is a function $P : \mathcal{G}(\mathcal{U}, \mathcal{I}) \to 2^{\mathcal{U}\times\mathcal{U}}$, such that $P(G) \subseteq V(G) \times V(G)$ for every social network $G \in \mathcal{G}(\mathcal{U}, \mathcal{I})$. Intuitively, a policy P specifies, for each social network G, a binary relation $P(G)$ that relates each resource owner to those accessors that the policy grants access. That is, $(u, v) \in P(G)$ iff owner u grants access to accessor v in the context of social network G. Note that $P(G)$ contains only finitely many pairs because G has only finitely many vertices. Using the notation defined in Section 2.1, we write $P(G)(u, v)$ whenever $(u, v) \in P(G)$.

In a typical ReBAC system, only a certain vocabulary of ReBAC policies is supported. For example, in Facebook, the policy vocabulary includes ReBAC policies such as me, friend, friend-of-friend, everyone, etc. A resource owner will adopt a supported policy from the policy vocabulary to protect a resource. The following are examples of ReBAC policies. They were previously proposed in [18] in the study of Facebook-style Social Network Systems (FSNSs). They have been adapted here as examples of ReBAC policies.

EXAMPLE 1 (DISTANCE). *Suppose the set of relational identifiers is $\mathcal{I} = \{\mathsf{friend}\}$. Suppose further that the relation identified by* friend *is symmetric (but not necessarily irreflexive). We call such social networks* ***pseudo-Facebook social networks*** *(the* friend *relation of a genuine Facebook-style social network is irreflexive). Define the policy dist_k such that $\mathsf{dist}_k(G)(u, v)$ iff u and v is at a distance k or less from one another. This policy is a generalization of the friend-of-friend policy in Facebook. Specifically, the distance between two vertices is taken as a measure of the strength of trust between two individuals. Having the trust level exceeding the threshold k results in access granting.*

[2]In [16], an access attempt also involves an access context. We ignore the latter as it is irrelevant to this work.

EXAMPLE 2 (COMMON FRIENDS). *Consider again pseudo-Facebook social networks. Define the policy cf_k such that $\mathsf{cf}_k(G)(u, v)$ iff (a) $u = v$, or (b) v is a neighbour of u, or (c) there are k (or more) common neighbours between u and v. Intuitively, access is granted to a stranger if k* friend*s of the owner witness to the trustworthiness of this stranger. This policy is another generalization of the friend-of-friend policy in Facebook. Specifically, the number of common neighbours is taken as a measure of the strength of trust between two strangers. Having the trust level exceeding the threshold k results in access granting.*

EXAMPLE 3 (CLIQUE). *Consider yet again pseudo-Facebook social networks. Define the policy clique_k such that $\mathsf{clique}_k(G)(u, v)$ iff (a) $u = v$, or (b) u and v belong to a clique*[3] *of size k (or more) in $\mathsf{acq}(G)$. Intuitively, access is granted if the accessor and the owner belong to a close-knit community of size k (or more). This policy is a generalization of the friend policy in Facebook. Specifically, the size of the largest common clique to which two neighbouring individuals belong is taken as a measure of the strength of trust between them. Having a trust level exceeding the threshold k results in access granting. Other notions of close-knit communities [9], such as clans, plexes and cores, can also be modelled in a similar way.*

2.4 Relational Policies

In an unpublished manuscript [1], Anwar *et al.* identify a family of policies, called relational policies, the use of which distinguishes ReBAC from traditional access control paradigms. Intuitively, a relational policy is special in two ways. First, a relational policy does not base authorization decisions on the identities of the owner or the accessor. Instead, only the topological structure the social network is analyzed for authorization. Second, a relational policy specifies how the owner and the accessor shall be related in the social network (e.g., the owner and the accessor are at a distance no more than k), rather than individual properties of the owner or the accessor (e.g., the accessor has a vertex degree no less than k). These two characteristics are captured in the following definitions, which we generalized from those in [1].

DEFINITION 4. *A policy P is* ***topology-based*** *iff, for every pair of isomorphic social networks $G, G' \in \mathcal{G}(\mathcal{U}, \mathcal{I})$ with isomorphism $\pi : V(G) \to V(G')$, we have $(u, v) \in P(G)$ whenever $(\pi(u), \pi(v)) \in P(G')$.*

In short, a topology-based policy does not take the identities of the owner and the accessor into consideration. Instead,

[3]A clique is a complete subgraph. Here we mean that u and v are part of a complete subgraph of the social network.

such a policy consumes only topological information of the social network.

The next definition characterizes policies that specify how the owner and the accessor are related to one another.

Definition 5. *A policy P is **local** iff, given $G \subseteq G'$ and $u, v \in V(G)$ for which $(u, v) \in P(G) \backslash P(G') \cup P(G') \backslash P(G)$, there exists $i \in \mathcal{I}$ such that $R_i(G') \backslash R_i(G)$ contains a pair (u', v') for which each of u' and v' is connected to each of u and v in G'.*

The intuition of the definition is the following. Suppose P is a local policy. Suppose further that the authorization decision of P is altered for a pair of individuals (u, v) after one adds vertices and edges into the social network G to obtain social network G'. Then G' must have received a new edge (u', v') of some relationship type i, such that each of u', v' is connected to each of u, v. In short, adding vertices or edges outside of the shared component of the owner and the accessor never alters the authorization decision of a local policy. Put more succinctly, the ends of every edge that has an influence on authorization decisions must connect both the owner and the accessor. That is, such an edge must contribute to the "connectedness" between the owner and the accessor. In this way, we capture the intuitive requirement that the policy must specify how the owner and the accessor are related to one another, rather than specifying individual graph properties of the owner or the accessor.

Definition 6. *A policy is **relational** iff it is both topology based and local.*

Note that the three example policies (i.e., distance, common friends and cliques) are relational policies.

2.5 The ReBAC Policy Language B

Fong proposed a policy language for expressing ReBAC policies [16]. We call his language B to differentiate it from the extended language E to be presented in the sequel. The language B is a basic modal logic [5]. Each formula in B expresses a ReBAC policy: i.e., a desirable relationship between two individuals in a social network. The syntax of a B formula, which is based on a set $\mathcal{I}$ of relation identifiers, is given below[4]:

$$\phi, \psi ::= \top \mid \mathsf{a} \mid \neg\phi \mid \phi \wedge \psi \mid \langle i \rangle\, \phi \mid \langle -i \rangle\, \phi$$

where $i \in \mathcal{I}$ is a relation identifier.

A policy (i.e., a formula) ϕ is interpreted in the context of a social network $G \in \mathcal{G}(V, \mathcal{I})$ and individuals $u, v \in V(G)$. Specifically, the semantics of B is given in a satisfiability relation $G, u, v \models_{\mathsf{B}} \phi$, which asserts that, in social network G, owner u and accessor v possess the relationship prescribed by policy ϕ.

- $G, u, v \models_{\mathsf{B}} \top$.
- $G, u, v \models_{\mathsf{B}} \mathsf{a}$ iff $u = v$.
- $G, u, v \models_{\mathsf{B}} \neg\phi$ iff it is not the case that $G, u, v \models_{\mathsf{B}} \phi$.
- $G, u, v \models_{\mathsf{B}} \phi \wedge \psi$ iff both $G, u, v \models_{\mathsf{B}} \phi$ and $G, u, v \models_{\mathsf{B}} \psi$.
- $G, u, v \models_{\mathsf{B}} \langle i \rangle\, \phi$ iff there exists $u' \in V(G)$ such that $(u, u') \in R_i(G)$ and $G, u', v \models_{\mathsf{B}} \phi$.
- $G, u, v \models_{\mathsf{B}} \langle -i \rangle\, \phi$ iff there exists $u' \in V(G)$ such that $(u', u) \in R_i(G)$ and $G, u', v \models_{\mathsf{B}} \phi$.

Intuitively, $\top$ is a relationship satisfiable by any pair of individuals; a asserts that the accessor is the owner; $\langle i \rangle\, \phi$ asserts that the owner has an i-neighbour that is related to the accessor in the manner specified by ϕ; $\langle -i \rangle\, \phi$ asserts that the owner is the i-neighbour of some individual that is related to the accessor in the manner specified by ϕ. The difference between $\langle i \rangle$ and $\langle -i \rangle$ is that the former traverses *along* the direction of an edge, while the latter traverses *against* the direction of an edge. Standard derived forms are defined:

$$\bot = \neg\top \qquad [i]\, \phi = \neg\langle i \rangle\, \neg\phi$$
$$\phi \vee \psi = \neg(\neg\phi \wedge \neg\psi) \qquad [-i]\, \phi = \neg\langle -i \rangle\, \neg\phi$$

Intuitively, $[i]\, \phi$ asserts that every i-neighbour of the owner is related to the accessor in the manner specified by ϕ.

A ReBAC policy P is ***definable in*** B iff there is a B formula ϕ such that $P(G)(u, v)$ whenever $G, u, v \models_{\mathsf{B}} \phi$. It has been demonstrated that many complex relational policies are definable in B. Specifically, Fong [16] was able to use B to capture all the trust delegation policies identified in the Electronic Health Records case study developed by Becker and Sewell [4]. We illustrate the usage of B with a few examples. Consider social networks with relation identifiers $\mathcal{I} = \{\mathsf{parent}, \mathsf{child}, \mathsf{spouse}, \mathsf{sibling}\}$, which signify respectively the child-parent relation, its inverse, the reflexive spouse-spouse relation, and the reflexive sibling-sibling relation. The following policies are definable in B:

"*Grant access to grand parents.*" The B formula that expresses this policy is "$\langle\mathsf{parent}\rangle\, \langle\mathsf{parent}\rangle\, \mathsf{a}$".

"*Grant access to a sibling who is not married.*" The B formula that expresses this policy is:

$$\langle\mathsf{sibling}\rangle\, (\mathsf{a} \wedge [\mathsf{spouse}]\, \bot)$$

"*Grant access if accessor is the only child of the owner.*" A formula to express the policy is:

$$\langle\mathsf{child}\rangle\, \mathsf{a} \wedge [\mathsf{child}]\, \mathsf{a}$$

2.6 Limitations of B

B is not without limitations. It turns out that some relational policies are not definable in B.

Example 7. *Assume the setting of Example 2. The policy cf_k, for every $k > 2$, is not definable in B. The reason is that the models for these policies are bisimilar to cf_1, and thus indistinguishable by a formula of B [5]. For instance,*

[4]The modal operator $\langle -i \rangle$ is not found in [16]. Instead, it is assumed in [16] that social networks are "inverse-closed." The two achieve the same effect: the authorization procedure can freely traverse either along or against the direction of an edge.

consider cf_2. A naive attempt to express cf_2 in B yields the following formula:

$$\mathsf{a} \vee \langle \mathsf{friend} \rangle \mathsf{a} \vee (\langle \mathsf{friend} \rangle \langle \mathsf{friend} \rangle \mathsf{a} \wedge \langle \mathsf{friend} \rangle \langle \mathsf{friend} \rangle \mathsf{a})$$

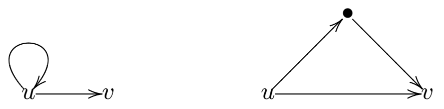
The intention is that the disjuncts "a" and "$\langle \mathsf{friend} \rangle \mathsf{a}$" respectively express requirements (a) and (b) in Example 2, while the third disjunct expresses requirement (c). Supposedly, each of the two conjuncts "$\langle \mathsf{friend} \rangle \langle \mathsf{friend} \rangle \mathsf{a}$" identifies a common friend of the owner and the accessor. Unfortunately, this formula does not express cf_2. Consider the scenario in which the owner and the accessor share exactly one common neighbour. Since boolean conjunction is idempotent, the formula above will be satisfied even though cf_2 should fail.

In the formula above, the original intention is that the two conjuncts specify two ***disjoint sets of intermediaries*** (i.e., two distinct common friends). Unfortunately, boolean conjunction fails to capture this requirement. This illustrates the need for an ReBAC policy language to provide a non-idempotent version of conjunction for specifying disjoint sets of intermediaries.

EXAMPLE 8. *The policy clique_k is not definable in B, for every $k > 2$. For instance, consider clique_3. A naive attempt to express the policy is the following formula:*

$$\mathsf{a} \vee (\neg \mathsf{a} \wedge \langle \mathsf{friend} \rangle \mathsf{a} \wedge \langle \mathsf{friend} \rangle (\neg \mathsf{a} \wedge \langle \mathsf{friend} \rangle \mathsf{a}))$$

The intention is that the disjunct "a" expresses requirement (a) in Example 3, while the second disjunct expresses requirement (b). In the second disjunct, we assert that the owner is not the accessor ($\neg \mathsf{a}$), the accessor is a friend of the owner ($\langle \mathsf{friend} \rangle \mathsf{a}$), and the accessor is a friend of a friend of the owner ($\langle \mathsf{friend} \rangle (\neg \mathsf{a} \wedge \langle \mathsf{friend} \rangle \mathsf{a})$). Yet, the following two social networks are not distinguishable by B formulas.



Specifically, the sub-formula "$\langle \mathsf{friend} \rangle (\neg \mathsf{a} \wedge \langle \mathsf{friend} \rangle \mathsf{a})$" fails to ensure that the owner (u) and the intermediary ($\bullet$) are two distinct vertices.

The above example illustrates an important shortcoming of the policy language B, namely, its lacking a ***vertex identification*** mechanism: i.e., the ability to name vertices and subsequently test if they are revisited.

3. THE REBAC POLICY LANGUAGE E

In the previous section, we identified two limitations of the policy language B, namely, disjoint intermediaries and vertex identification. In the following, we will look at how the two features can be incorporated into a ReBAC policy language. The result of this exploration is an extended ReBAC policy language we call E.

Syntax. As before, E is defined in the context of a countable set $\mathcal{U}$ of vertices and a countable set $\mathcal{I}$ of relation identifiers. We also assume that there is a countably infinite set $\mathcal{P}$ of propositional symbols. We write p and q for typical members of $\mathcal{P}$. We also assume that $\mathcal{P}$ contains a distinguished member a.

The syntax of E is given below:

$$\phi, \psi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \psi \mid \langle i \rangle \phi \mid \langle -i \rangle \phi \mid @p.\phi \mid \phi \otimes \psi$$

where $p \in \mathcal{P}$ is a propositional symbol, and $i \in \mathcal{I}$ is a relation identifier. Intuitively, $@p.\phi$ introduces a propositional symbol for identifying the owner. Specifically, the proposition symbol p can be used inside ϕ to test if a vertex is the owner. The formula $\phi \otimes \psi$ holds whenever the relationships prescribed by ϕ and ψ both hold between the accessor and the owner, but the set of intermediary vertices used for establishing these two relationships are disjoint. A more precise definition of the semantics of these new constructs is given below.

The notion of ***free*** and ***bound*** occurrences of propositional symbols can be defined in a standard way. For example, in the formula $(p \wedge @q.\langle i \rangle q)$, the propositional symbol p occurs free but q occurs bound. A formula is a-***closed*** if no propositional symbol other than a occurs free in the formula. Otherwise the formula is a-***open***.

Semantics. Given a social network G, a function $\Sigma : \mathcal{P} \rightharpoonup V(G)$ is called a ***binding environment***[5]. A binding environment interprets a propositional symbol as a vertex.

The semantics of E is defined via the satisfaction relation $G, u, v \models_{\mathsf{E}} \phi$, which is specified in terms of an auxiliary relation $G, \Sigma, u \Vdash \phi$, where Σ is a binding environment.

- $G, \Sigma, u \Vdash \top$.
- $G, \Sigma, u \Vdash p$ iff $\Sigma(p) = u$.
- $G, \Sigma, u \Vdash \neg\phi$ iff it is not the case that $G, \Sigma, u \Vdash \phi$.
- $G, \Sigma, u \Vdash \phi \wedge \psi$ iff both $G, \Sigma, u \Vdash \phi$ and $G, \Sigma, u \Vdash \psi$.
- $G, \Sigma, u \Vdash \langle i \rangle \phi$ iff there exists $u' \in V$ such that $(u, u') \in R_i$ and $G, \Sigma, u' \Vdash \phi$.
- $G, \Sigma, u \Vdash \langle i \rangle \phi$ iff there exists $u' \in V$ such that $(u', u) \in R_i$ and $G, \Sigma, u' \Vdash \phi$.
- $G, \Sigma, u \Vdash @p.\phi$ iff $p \neq \mathsf{a}$ and $G, \Sigma[p \mapsto u], u \Vdash \phi$.
- $G, \Sigma, u \Vdash \phi \otimes \psi$ iff the following holds:

  Let v be $\Sigma(\mathsf{a})$. There exists two subsets V_1 and V_2 of $V(G)$, such that $V(G) = V_1 \cup V_2$, $V_1 \cap V_2 = \{u, v\}$, and both $G[V_1], \Sigma, u \Vdash \phi$ and $G[V_2], \Sigma, u \Vdash \phi$ hold.

Lastly, $G, u, v \models_{\mathsf{E}} \phi$ iff $G, \emptyset[\mathsf{a} \mapsto v], u \Vdash \phi$.

[5]Note that our binding environment plays the role of a valuation or labelling function in the standard literature of modal logic [5], in which a valuation is a function with signature $\mathcal{P} \to 2^{V(G)}$, and a labelling function is a function with signature $V(G) \to 2^{\mathcal{P}}$. In our logic, a propositional symbol is interpreted as a single vertex. We therefore adapt the definition to ease presentation.

Derived Forms. We define the derived forms $\bot$, $\phi \vee \psi$, $[i]\,\phi$ and $[-i]\,\phi$ as before. In addition, we introduce the following derived form:

$$\phi \oplus \psi = \neg(\neg\phi \otimes \neg\psi)$$

The connective $\oplus$ is the dual of $\otimes$. Specifically, $G, \Sigma, u \Vdash \phi \oplus \psi$ iff the following holds:

> Let v be $\Sigma(\mathsf{a})$. For every two subsets V_1 and V_2 of $V(G)$ such that $V(G) = V_1 \cup V_2$, $V_1 \cap V_2 = \{u, v\}$, either $G[V_1], \Sigma, u \Vdash \phi$ or $G[V_2], \Sigma, u \Vdash \phi$ holds.

In other words, $G, \Sigma, u \Vdash \phi \oplus \psi$ whenever, for every subset V of $V(G)$ such that $\{u, \Sigma(\mathsf{a})\} \subseteq V$, either $G[V], \Sigma, u \Vdash \phi$ or $G[V], \Sigma, u \Vdash \phi$ holds.

Definability. A policy P is ***definable in*** E iff there is an a-closed formula ϕ in E such that $P(G)(u, v)$ whenever $G, u, v \models_{\mathsf{E}} \phi$.

Examples. The following examples illustrate how the new features of E address the needs for disjoint intermediaries and vertex identification.

EXAMPLE 9. *The policy cf_2 is definable in E, via the following formula:*

$$\mathsf{a} \vee \langle \mathsf{friend} \rangle\, \mathsf{a} \vee ((\langle \mathsf{friend} \rangle \langle \mathsf{friend} \rangle\, \mathsf{a}) \otimes (\langle \mathsf{friend} \rangle \langle \mathsf{friend} \rangle\, \mathsf{a}))$$

EXAMPLE 10. *The policy clique_3 is definable in E, via the following formula:*

$$\mathsf{a} \vee (\neg\mathsf{a} \wedge \langle \mathsf{friend} \rangle\, \mathsf{a} \wedge @p.\langle \mathsf{friend} \rangle\, (\neg p \wedge \neg\mathsf{a} \wedge \langle \mathsf{friend} \rangle\, \mathsf{a}))$$

4. EXPRESSIVENESS OF E

It has been pointed out in Section 2.4 that ReBAC is distinguished by its extensive use of access control policies are *relational*: i.e., authorization decisions consume only topological information of the social network (topology-based), and express how the owner and the accessor are related in the social network (local). In this section, we will examine the expressiveness of E by identifying what family of relational policies are definable in E. We will first establish, in Section 4.1, an "upper bound" of the policies definable in E: i.e., identifying a natural family of ReBAC policies to which every E-definable policy belongs. We will then establish, in Section 4.2, a "lower bound" of the expressiveness of E. That is, we will identify a natural subclass of relational policies that are definable in E. In other words, we establish the representational completeness of E with respect to that subclass of relational policies. We conclude the section with a structural analysis of the hierarchy of policy families identified along the way (Section 4.3).

4.1 Owner-checkable policies

We begin our discussion with the family of topology-based policies that can be enforced by an agent that traverses a neighbourhood of the owner.

DEFINITION 11. *A topology-based policy P is **owner-checkable (OC)** iff $P(G)(u, v) \Leftrightarrow P(C_G(u; v))(u, v)$. A topology-based policy P is **accessor-checkable (AC)** iff $P(G)(u, v) \Leftrightarrow P(C_G(v; u))(u, v)$.*

Intuitively, the authorization decision of an OC policy can be determined by examining *only* the component of social network in which the owner is located, plus the additional knowledge of whether the accessor is in that component ($C_G(u; v)$). That is, even if there may be vertices and edges outside of the owner's component, they never influence the authorization decision. The intuition of an AC policy is analogous.

An OC policy is special in two ways. Firstly, an OC policy presents a tractability advantage. In particular, an OC policy can be evaluated by a "crawler" that traverses only the owner's component (i.e., neighbourhood), starting at the owner vertex. Venturing outside of the owner's component is never necessary. An AC policy shares the same tractability advantage if the traversal begins at the accessor's vertex. Secondly, an OC policy provides a security advantage. Although an AC policy provides the same tractability advantage as an OC policy, only an OC policy provides this form of security advantage. The checking of an AC policy can be conducted by a "crawler" that traverses the accessor's component, starting at the accessor. Yet, an AC policy may lead to a denial-of-service attack by the accessor. Specifically, the accessor may carefully craft its neighbourhood in such a way that would lead the "crawler" (i.e., the authorization procedure) to explore a graph neighbourhood that is expensive to traverse (with respect to the policy in question). Even if access is not granted, an accessor may repeatedly request access, leading the authorization procedure to waste precious computational resources. An OC policy prevents this form of manipulation, in the sense that the accessor cannot dictate the computational cost of authorization.

Relational policies are related to OC and AC policies.

PROPOSITION 12. *A relational policy is both OC and AC.*

(A proof of this proposition can be found in Appendix A.)

The following theorem represents an "upper bound" to the expressiveness of E.

THEOREM 13. *A policy definable in E is OC.*

(A proof of this theorem can be found in Appendix A.

4.2 Finitary Relational Policies

We now turn to the "lower bound" of E's expressiveness. The authorization decisions of a ReBAC policy can be determined in two ways. One is to grant access when certain relationships are present in the social network, the other is to grant access when certain relationships are absent. The evaluation of a policy of the second kind requires complete knowledge of the entire social network, while the evaluation of a policy of the first kind requires only a fragment of the social network to provide an existential proof of compliance. The following definition formalizes the idea.

Definition 14. *A policy P is **monotonic** iff $G \subseteq G'$ implies $P(G) \subseteq P(G')$.*

That is, adding relationships into a social network never reduces accessibility, and removing relationships never expands accessibility. Such a policy makes authorization decisions by verifying the presence of relationships rather than the absence of relationships. The notion of monotonicity was originally proposed in [18] for Facebook-style Social Network Systems. Here we generalize the notion for ReBAC policies.

Adopting policies that are exclusively monotonic enables a decentralized implementation of ReBAC in the style of trust management systems [6, 40, 27, 26]. Specifically, an accessor who seeks authorization may present to the reference monitor a fragment of the social network (e.g., a collection of certificates of relationships) as a proof of compliance. ReBAC, however, differs from trust management systems in its extensive use of relational policies, which do not rely on the identity of an individual for making authorization decision. We are therefore interested in policies that are both monotonic and topology based.

Definition 15. *A policy P is **positive** iff it is both topology based and monotonic.*

We review here a characterization of positive policies. Our characterization is phrased in terms of birooted graphs.

Definition 16. *A **birooted graph** $G_{(u,v)}$ is a triple $\langle G, u, v\rangle$ such that G is a social network and $u, v \in V(G)$. The vertices u and v are the **roots** of $G_{(u,v)}$, and they need not be distinct. Specifically u is the **owner root**, and v is the **accessor root**. We write $\mathcal{B}(S, \mathcal{I}) = \{G_{(u,v)} \mid G \in \mathcal{G}(S, \mathcal{I}),\ u, v \in V(G)\}$ to denote the set of all birooted graphs based on vertex set S. ${G'}_{(u,v)}$ is a **(birooted) subgraph** of $G_{(u,v)}$, written as ${G'}_{(u,v)} \subseteq G_{(u,v)}$, iff $G' \subseteq G$. (Note the matching roots.) We say that $G_{(u,v)}$ and ${G'}_{(u',v')}$ are **isomorphic** iff there exists an isomorphism $\pi : V(G) \to V(G')$ between social networks G and G', such that $\pi(u) = u'$ and $\pi(v) = v'$. In this case we write $G_{(u,v)} \cong {G'}_{(u',v')}$. We also write ${G'}_{(u',v')} \lesssim G_{(u,v)}$ whenever there is a birooted graph ${G''}_{(u,v)}$ such that ${G'}_{(u',v')} \cong {G''}_{(u,v)}$ and ${G''}_{(u,v)} \subseteq G_{(u,v)}$.*

We are now ready to state the characterization theorem for positive policies.

Definition 17. *The policy **positively induced** by a set $\mathcal{B}$ of birooted graphs is the policy $P^+_{\mathcal{B}}$ for which*

$$P^+_{\mathcal{B}}(G)(u,v) \text{ iff } \exists {G'}_{(u',v')} \in \mathcal{B} \,.\, {G'}_{(u',v')} \lesssim G_{(u,v)}$$

Intuitively, the birooted graph set $\mathcal{B}$ specifies topological "patterns" that must exist between an owner and an accessor in the social network in order for access to be granted by the policy. It was proven in [1] that every positive policy can be characterized by a set of birooted graph patterns. The result was originally established for Facebook-style Social Network Systems. Here, we adapt the result to the more general context of ReBAC.

Theorem 18. *Every positive policy is positively induced by a set of birooted graphs. The minimal set of birooted graphs to positively induce a given policy P is defined to be the set $\mathcal{B}$ for which there exists no proper subset of $\mathcal{B}$ that also positively induces P. This minimal set does not contain a pair of distinct birooted graphs $G_{(u,v)}$ and ${G'}_{(u',v')}$ such that $G_{(u,v)} \lesssim {G'}_{(u',v')}$. Such a minimal set always exists, and is unique up to birooted graph isomorphism.*

Intuitively, the theorem states that every positive policy can be uniquely characterized by a smallest set of birooted graph patterns.

We are particularly interested in a special kind of positive policies.

Definition 19. *A positive policy P is **finitary** iff the minimal set of birooted graphs to positively induce P is a finite set.*

Finitary policies can be characterized by a finite number of birooted graphs. While positive policies can be evaluated by checking for the presence of relationships, the finitary requirement ensures that only a bounded number of such relationships need to be enumerated. This requirement is computationally significant in two ways. Firstly, this translates to a bound in the search effort for a centralized implementation of the authorization procedure, for the search tree now has bounded depth. Secondly, in a distributed implementation of ReBAC, this constraint implies that a proof of compliance consists of a bounded number of certificates.

We are now ready to state the main theorem of this work.

Theorem 20. *Every finitary OC policy is definable in* E.

The theorem provides a "lower bound" of what can be expressed in E. The proof of this theorem can be found in Appendix A.

Recall that our core interest is in expressing relational policies, rather than OC policies. The following corollary, which follows directly from proposition 12, identifies a subclass of relational policies definable in E.

Corollary 21. *Every finitary relational policy is definable in* E.

Note that all the three example policies (i.e., distance, common friends, and clique) are finitary relational policies.

We can further strengthen the above result. Suppose P_1 and P_2 are definable in E via the formulas ϕ_1 and ϕ_2. Consider

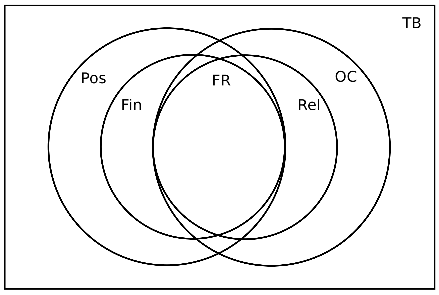


Figure 1: The hierarchy of policy families identified in this work: TB: topology-based policies; OC: owner-checkable policies; Rel: relational policies; Pos: positive policies; Fin: finitary policies; FR: finitary relational policies.

the policy P defined such that $P(G) = P_1(G) \cap P_2(G)$. That is, P grants access whenever both P_1 *and* P_2 grant access. It is easy to see that P is also definable in E, via the formula $\phi_1 \wedge \phi_2$. The same can be said about other boolean combinations (disjunction and negation).

COROLLARY 22. *The family of policies definable in* E *is closed under boolean combinations. Consequently, boolean combinations of finitary relational policies are definable in* E. *Note that such boolean combinations of finitary relational policies are still relational policies.*

This corollary gives us an easy extension of the definability result in Corollary 21.

4.3 Discussion

Figure 1 depicts the hierarchy of policy families identified in this section. Note that *every inclusion in Figure 1 is proper.* To demonstrate this, the following list states a member policy for each set difference.

$\mathsf{TB} \setminus (\mathsf{Pos} \cup \mathsf{OC})$: The social network contains no more than k vertices (including the owner and the accessor).

$\mathsf{Pos} \setminus \mathsf{Fin}$: The owner and the accessor are connected.

$\mathsf{Fin} \setminus \mathsf{FR}$: The accessor has at least k neighbours.

$\mathsf{OC} \setminus \mathsf{Rel}$: The owner has no more than k neighbours.

$\mathsf{Rel} \setminus \mathsf{FR}$: The owner and the accessor are not connected.

5. RELATED WORK

The term Relationship-Based Access Control (ReBAC) was first coined by Gates [19] when she articulated the need for an access control paradigm that is based on interpersonal relationships. A number of ReBAC models and/or systems have been proposed for social computing applications [25, 8, 7, 37, 38, 18]. The targets of study include the distributed evaluation of trust [8], the employment of semantic web technology to encode social networks and ReBAC policies [7], the automatic generation of ReBAC policies [37], multiple ownership of resources [38], the generalization of Facebook-style Social Network Systems [18], and the characterization of relational policies [1]. Fong advocates the employment of ReBAC in application domains outside of social computing [16], and demonstrates the utility and feasibility of doing so by the specification of a general-purpose ReBAC model and a ReBAC policy language. He points out that the various constructs in his ReBAC model arise naturally out of the generalization of foundational concepts such as roles, role hierarchy, sessions and constraints in Role-Based Access Control (RBAC) [33, 13]. His work also raises the question of representational completeness, which is the focus of this work. The core contribution of this work is (a) demonstrating that Fong's policy language fails to encode a number of relational policies previously studied in [18], (b) incorporating the support of disjoint intermediaries and vertex identification into a ReBAC policy language that is based on modal logic, and (c) providing both an upper bound and a lower bound to the family of policies expressible in the extended policy language.

The question of representational completeness is related to enforceability research [36, 29, 14, 23, 39, 30, 31]. The goal of enforceability research is to establish the structural relationships between naturally occurring policy families on one hand (e.g., safety policies), and the family of policies enforceable by a specific kind of enforcement mechanisms (e.g., program monitors that have access only the execution history). Rather than relating policy families to *enforcement mechanisms*, the present work establishes upper bound and lower bound for the family of policies expressible in a *policy language*.

Policies such as cf_k (for $k > 2$) are not definable in B. The reason is that the models for these policies are bisimilar to cf_1, and thus indistinguishable by a formula of B [5]. Therefore, in E a new connective $\otimes$ is introduced to require disjoint partitioning of the social network during a satisfiability check (except for the accessor and the owner). This approach is similar to *manifold roles* by Li *et al.* in RT^T [27]. The credential using $\otimes$ in RT^T is a union of the sets where the intersection of such sets is empty. The concept of *threshold structure* by Li *et al.* in Delegation logic [26] is also similar to $\otimes$. In a threshold structure, k out of N principals are required, where each of the k is unique.

6. FUTURE WORK

Some extensions to the ReBAC model have been suggested in [16]. We discuss in the following future directions that are specific to ReBAC policy languages.

A natural future work is to improve the precision of the characterization of policies definable in E. Specifically, there is a gap between the "upper bound" and "lower bound" presented in Section 4. As demonstrated in Section 4.3, the inclusion of FR in OC is proper. It is desirable if one can close the gap, and provide a precise identification of the family of policies definable in E. We anticipate that such an effort can benefit from the application of techniques from Finite Model Theory [28], which has long been applied to understand the expressiveness of database query languages [11, 10, 34].

One of the motivations for designing a policy language for ReBAC policies is to facilitate policy analysis. It has been shown [15] that proper static analysis can be applied to the configuration of Facebook-style Social Network Systems to ensure the absence of Sybil attacks. It is interesting to explore if this or other policy analyses can be translated into an equivalent syntactic analysis for policies written in a ReBAC policy language.

Both this work and [16] assumes the ReBAC system tracks user relationships explicitly, and the relations are in turn identified by a fixed vocabulary of relation identifiers. This assumption has made the design of rich policy languages possible. In some applications, however, the presence of a relationship edge as well as its typing are induced by complex external factors. For example, consider a network consisting of users and devices such as mobile phones, in which the presence of a relationship is the result of the states of the mobile phones (e.g., battery level) and their relative proximity. Rational modelling of such external dependence would allow us to reuse ReBAC and our policy languages even in these application domains.

Another future direction is to combine the strengths of both RBAC and ReBAC, thereby allowing the access control system to reason about both the relationships between individuals, and the hierarchical organization of user roles. One possibility is to employ Description Logic (DL) technology [2] as the basis of both the policy language and the authorization engine. Highly related to modal logics [35], DL is a mature technology, with language families of various tractability, and the support of efficient reasoners (e.g., RACER [22] and FaCT [24]). A previous work on applying DL to access control is that of RelBAC [20, 41], in which DL is employed to construct an access control model akin to the Domain-Type Enforcement (DTE) model [3]. Further work is needed to enable the rational marriage of ReBAC and RBAC in one model.

7. CONCLUSION

In this work, we examined the relative expressiveness of two ReBAC policy languages. We began by demonstrating that there are known relational policies, such as cf_k and clique_k, that cannot be expressed in the ReBAC policy language B. We pointed out that the limitations were due to the lack of support for disjoint intermediaries and vertex identification. The two features were then incorporated into B, resulting in an extended policy language E. We showed that ReBAC policies definable in E are owner-checkable policies, a superset of relational policies. We also showed that every finitary relational policy is definable in E, meaning that E is representationally complete with respect to that policy family. We have therefore identified a policy language that is representationally adequate for expressing useful ReBAC policies.

Acknowledgements

This work is supported in part by an NSERC Discovery Accelerator Supplements.

8. REFERENCES

[1] Mohd Anwar, Zhen Zhao, and Philip W. L. Fong. An access control model for Facebook-style social network systems. Technical Report 2010-959-08, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, July 2010.

[2] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge, 2007.

[3] Lee Badger, Daniel F. Sterne, David L. Sherman, Kenneth M. Walker, and Sheila A Haghighat. Practical domain and type enforcement for UNIX. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pages 66–77, Oakland, CA, USA, May 1995.

[4] Moritz Y. Becker and Peter Sewell. Cassandra: Flexible trust management, applied to electronic health records. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, Pacific Grove, CA, USA, June 2004.

[5] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge, 2001.

[6] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy (S&P'96)*, pages 164–173, Oakland, CA, USA, May 1996.

[7] Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thuraisingham. A semantic web based framework for social network access control. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies (SACMAT'09)*, pages 177–186, Stresa, Italy, June 2009.

[8] Barbara Carminati, Elena Ferrari, and Andrea Perego. Enforcing access control in Web-based social networks. *ACM Transactions on Information and System Security*, 13(1):1–38, October 2009.

[9] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38, March 2006.

[10] Ashok K. Chandra. Theory of database queries. In *Proceedings of the Seventh ACM Symposium on Principles of Database Systems (PODS'88)*, pages 1–9, Austin, Texas, USA, 1988.

[11] Ashok K. Chandra and David Harel. Computable queries for relational data bases. *Journal of Computer and System Sciences*, 21(2):156–178, October 1980.

[12] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[13] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, August 2001.

[14] Philip W. L. Fong. Access control by tracking shallow execution history. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy (S&P'04)*, pages 43–55, Oakland, CA, USA, May 2004.

[15] Philip W. L. Fong. Preventing Sybil attacks by privilege attenuation: A design principle for social network systems. Technical Report 2010-984-33, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, November 2010.

[16] Philip W. L. Fong. Relationship-based access control: Protection model and policy language. In *Proceedings*

of the First ACM Conference on Data and Application Security and Privacy (CODASPY'11), San Antonio, Texas, USA, February 2011. To appear. A preliminary version of the paper is available as [17].

[17] Philip W. L. Fong. Relationship-based access control: Protection model and policy language. Technical Report 2010-974-23, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, September 2011.

[18] Philip W. L. Fong, Mohd Anwar, and Zhen Zhao. A privacy preservation model for Facebook-style social network systems. In *Proceedings of the 14th European Symposium on Research In Computer Security (ESORICS'09)*, volume 5789 of *Lecture Notes in Computer Science*, pages 303–320, Saint Malo, France, September 2009. Springer.

[19] Carrie E. Gates. Access Control Requirements for Web 2.0 Security and Privacy. In *IEEE Web 2.0 Privacy and Security Workship (W2SP'07)*, Oakland, CA, USA, 2007.

[20] Fausto Giunchiglia, Rui Zhang, and Bruno Crispo. RelBAC: Relation based access control. In *Proceedings of the Fourth International Conference on Semantics, Knowledge and Grid (SKG'08)*, pages 3–11, Beijing, China, December 2008.

[21] Luigi Giuri and Pietro Iglio. Role templates for content-based access control. In *Proceedings of the Second ACM Workshop on Role-Based Access Control (RBAC'97)*, pages 153–159, Fairfax, Virginia, USA, November 1997.

[22] Volker Haarslev and Ralf Möller. RACER system description. In *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR'01)*, pages 701–705, Siena, Italy, 2001. Springer.

[23] Kevin W. Hamlen, Greg Morrisett, and Fred B. Schneider. Computability classes for enforcement mechanisms. *ACM Transactions on Programming Langanguages And Systems*, 28(1):175–205, January 2006.

[24] Ian R. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–649, Trento, Italy, 1998.

[25] Sebastian Ryszard Kruk, Slawomir Grzonkowski, Adam Gzella, Tomasz Woroniecki, and Hee-Chul Choi. D-FOAF: Distributed identity management with access rights delegation. In *Proceedings of the First Asian Semantic Web Conference (ASWC'06)*, volume 4185 of *Lecture Notes in Computer Science*, pages 140–154, Beijing, China, September 2006. Springer.

[26] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security*, 6(1):128–171, February 2003.

[27] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P'02)*, pages 114–130, Berkeley, California, USA, May 2002.

[28] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[29] Jay Ligatti, Lujo Bauer, and David Walker. Edit automata: Enforcement mechanisms for run-time security policies. *International Journal of Information Security*, 4(1–2):2–16, February 2005.

[30] Jay Ligatti, Lujo Bauer, and David Walker. Run-time enforcement of nonsafety policies. *ACM Transactions on Information and Systems Security*, 12(3), January 2009.

[31] Jay Ligatti and Srikar Reddy. A theory of runtime enforcement, with results. In *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS'10)*, volume 6345 of *Lecture Notes in Computer Science*, Athens, Greece, September 2010. Springer.

[32] Emil Lupu and Morris Sloman. Reconciling role based management and role based access control. In *Proceedings of the Second ACM Workshop on Role-Based Access Control (RBAC'97)*, pages 135–141, Fairfax, Virginia, USA, November 1997.

[33] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 19(2):38–47, February 1996.

[34] Peter Schäuble and Beat Wüthrich. On the expressive power of query languages. *ACM Transactions on Information Systems*, 12:69–91, January 1994.

[35] Klaus Schild. A correspondence theory for terminological logics: preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial intelligence (IJCAI'91)*, pages 466–471. Morgan Kaufmann, 1991.

[36] Fred B. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, 2000.

[37] Anna Squicciarini, Federica Paci, and Smitha Sundareswaran. PriMa: An effective privacy protection mechanism for social networks. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS'10)*, pages 320–323, Beijing, China, April 2010.

[38] Anna C. Squicciarini, Mohamed Shehab, and Joshua Wede. Privacy policies for shared content in social network sites. *The VLDB Journal*, 2010. To appear.

[39] Chamseddine Talhi, Nadia Tawbi, and Mourad Debbabi. Execution monitoring enforcement under memory-limitation constraints. *Information and Computation*, 206:158–184, 2008.

[40] Stephen Weeks. Understanding trust management systems. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy (S&P'01)*, pages 94–105, Oakland, California, USA, May 2001.

[41] Rui Zhang, Fausto Giunchiglia, Bruno Crispo, and Lingyang Song. Relation-based access control: An access control model for context-aware computing environment. *Wireless Personal Communications*, 55(1):5–17, September 2010.

APPENDIX

A. PROOF

Proposition 12. *A relational policy is both OC and AC.*

Proof. Suppose P is a relational policy. We show that P is OC. (The proof of the assertion that P is also AC is symmetrical.) Consider the social network G and vertices $u, v \in V(G)$. It is obvious that $C_G(u; v) \subseteq G$. There is no $i \in \mathcal{I}$ such that $R_i(G) \backslash R_i(C_G(u; v))$ contains a pair (u', v') for which each of u' and v' is connected to u in G. Consequently, by Definition 5, $P(C_G(u; v)) \backslash P(G) \cup P(G) \backslash P(C_G(u; v))$ is an empty set, and thus $P(G)(u, v) \Leftrightarrow P(C_G(u; v))(u, v)$. □

Theorem 13. *A policy definable in* E *is OC.*

Proof. Had it not been for the construct $@p.\phi$, this theorem could have been proven in a straightforward manner via structural induction on the abstract syntax tree of a formula. The subtlety arises from the fact that ϕ may contain free occurrences of variables.

To address the presence of free variables, we prove a claim regarding a pair (Σ, ϕ), where Σ is an environment, and ϕ is a formula. A formula ϕ is said to be Σ-***closed*** iff every free variable of ϕ belongs to the domain of Σ. A pair (Σ, ϕ) is a ***closure*** iff ϕ is Σ-closed. We also write $\pi(\Sigma)$ to denote the environment Σ' defined over the same domain as Σ, such that $\Sigma'(p) = \pi(\Sigma(p))$.

Claim: Given a closure (Σ, ϕ) such that $\Sigma(\mathsf{a})$ is defined, the following conditions hold:

1. Let π be an isomorphism between social networks G and G'. Then $G, \Sigma, u \Vdash \phi$ iff $G', \pi(\Sigma), \pi(u) \Vdash \phi$.
2. $G, \Sigma, u \Vdash \phi$ iff $C_G(u; \Sigma(\mathsf{a})), \Sigma, u \Vdash \phi$.

The theorem follows directly from the above claim. The proof of the claim is a straightforward structural induction on the abstract syntax tree of the formula in the closure. □

The minimal set of birooted graphs to positively induce an OC policy has a special property that we will rely on in the proof of Theorem 20.

Lemma 23. *Suppose P is both positive and OC. Let $\mathcal{B}$ be the minimal set of birooted graphs to positively induce P. For every $G_{(u,v)} \in \mathcal{B}$, every vertex in $V(G) \backslash \{v\}$ is connected to u.*

The proof of the lemma is elementary.

Theorem 20. *Every finitary OC policy is definable in* E.

Proof. We describe a sketch of the proof.

Suppose P is a finitary OC policy. As P is positive, let $\mathcal{B}$ be the minimal set of birooted graphs to positively induce P. Since P is finitary, $\mathcal{B}$ is finite. We outline in the following the construction of an E formula ϕ for each birooted graph $G_{(u,v)} \in \mathcal{B}$ such that, for every $G'_{(u',v')} \in \mathcal{B}(U, \mathcal{I})$, $G_{(u,v)} \lesssim G'_{(u',v')}$ implies $G', u', v' \models_{\mathsf{E}} \phi$. The formula required by the theorem is the disjunction of the constructed formulas (recall that there are only finitely many of such formulas as $\mathcal{B}$ is finite).

Given birooted graph $G_{(u,v)} \in \mathcal{B}$, we construct a corresponding formula $\phi(u)$ as follows:

Step 1: We label each vertex by a distinct propositional symbol. (Since we have countably infinitely many propositional symbols, we always have enough symbols to work with.) In the following, we write p_x for the propositional symbol associated with a vertex x.

Step 2: Since P is both positive and OC, Lemma 23 implies that the vertices in G (perhaps with the exception of v) are connected to u. With u as the root, one can therefore build a depth-first search tree T. We assume that the search algorithm may traverse either along or against the direction of an edge. That is, a tree edge linking a parent vertex and a child vertex may go in either direction. Following standard terminology, we call the non-tree edges in G ***back edges***. A back edge always link a descendent vertex with an ancestor vertex. (Again, a back edge may either point from a descendent vertex to an ancestor vertex, or point the other direction.) For more details regarding DFS trees, consult a standard text such as [12]. Note that v may or may not be part of T, but all other vertices are in T. Also, every edge in G is either a tree edge or a back edge.

Step 3: For each vertex x in search tree T, construct a formula $\phi(x) = \psi_1 \wedge \psi_2 \wedge @p_x.(\psi_3 \wedge \psi_4)$, where:

- If x is not the root of search tree T (i.e., $x \neq u$), then $\psi_1 = \neg p_{x_1} \wedge \neg p_{x_2} \wedge \ldots \wedge p_{x_k}$, where $x_1, x_2, \ldots, x_k$ are the proper ancestors of x (i.e., excluding x itself). In case $x = u$, then $\psi_1 = \top$.
- If $x = v$, then $\psi_2 = \mathsf{a}$. Otherwise, $\psi_2 = \neg\mathsf{a}$.
- If x is a leaf of search tree T, then $\psi_3 = \top$. Otherwise, a subformula is constructed for each child of x in T. Suppose $(x, y) \in R_i(G)$ is a tree edge linking x with its child y in T. The subformula corresponding to y is $\langle i \rangle\, \phi(y)$. If the tree edge points in the other direction (i.e., $(y, x) \in R_i(G)$), then we use $\langle -i \rangle$ instead. Let ψ_3^1, ψ_3^2, …, ψ_3^m be the subformulas constructed in this way. We construct $\psi_3 = \psi_3^1 \otimes \psi_3^2 \otimes \ldots \otimes \psi_3^m$.
- If x is not incident on a back edge in G, then $\psi_4 = \top$. Otherwise, ψ_4 is a conjunction of formulas, one for each back edge linking x with one of its ancestors. Suppose $(x, y) \in R_i(G)$ is one such back edge. The conjunct corresponding to this back edge is $\langle i \rangle\, p_y$. If the back edge points in the other direction (i.e., $(y, x) \in R_i(G)$), then we use $\langle -i \rangle$ instead.

The construction can proceed recursively, with $\phi(y)$ constructed prior to $\phi(x)$ whenever y is a child of x. The recursion terminates properly because T is a tree of finite size. The intuition behind the construction is that $\phi(u)$ is an encoding of G, including both the subgraph T and all the back edges. More precisely, $\phi(u)$ encodes a DFS that an authorization procedure can deploy to confirm that a birooted graph contains a subgraph isomorphic to the “pattern” $G_{(u,v)}$. The detailed proof of this last claim is mechanical. $\square$