Relationship-Based Access Control: Its Expression and Enforcement Through Hybrid Logic

ABSTRACT

Access control policy is typically defined in terms of attributes, but in many applications it is more natural to define permissions in terms of relationships that resources, systems, and contexts may enjoy. The paradigm of relationship-based access control has been proposed to address this issue, and modal logic has been used as a technical foundation.

We argue here that hybrid logic – a natural and well-established extension of modal logic – addresses limitations in the ability of modal logic to express certain relationships. Also, hybrid logic has advantages in the ability to efficiently compute policy decisions relative to a relationship graph.

We identify a fragment of hybrid logic to be used for expressing relationship-based access-control policies, show that this fragment supports important policy idioms, and study its expressiveness. We also capture the previously studied notion of *relational* policies in a static type system.

Finally, we point out that use of our hybrid logic removes an exponential penalty in existing attempts of specifying complex relationships such as "at least three friends".

1. INTRODUCTION

Access control is typically specified and enforced in terms of attributes: authenticated properties that the resource, its system or context must possess in order to grant an access request. For example, one may express and enforce an access-control policy that, during weekends, managers may read company email whilst being connected through a virtual private network outside of company premises.

But there are many applications in which the decision of granting access should not primarily be based on attributes (e.g. whether a VPN connection is on or off) but rather on relationships that resources, systems, and contexts may enjoy. For example, a teenager may want to share pictures from a concert only with friends who actually went to the event. Expressing such policies through attributes is hard to do even within a monolithic and closed system, and is simply not feasible in distributed and open systems.

The paradigm of relationship-based access control [15, 12, 10] has been proposed to address this shortcoming of attributes. This research gives us a first understanding of appropriate *semantic* notions for relationship-based access control (see e.g. [10]). Yet, despite the initial progress reported in [12], it is less clear what appropriate syntactic counterparts these policies should have.

Ideally, a policy language for relationalship-based access control should be

- expressive enough to capture important policy idioms
- not so expressive as to make reasoning intractable
- intuitive for expressing and enforcing access control,
- formal, to support policy analysis, implementation, and optimization, and
- based on robust mathematical foundations.

Recent work [12] has proposed the use of modal logics as such a mathematical foundation. We entirely agree with the spirit of that proposal: a policy should be specified as a formula of some logic. But the work in [12] already recognized that modal logic alone cannot express some pertinent relationships. Features that appear to be lacking include:

- the ability to bind a node to a principal in a relationship graph
- graded variants of modalities, e.g., "at least four friends"
- the ability to evaluate sub-policies from the perspective of a named principal, and
- the ability to *efficiently* compute a policy decision by evaluating a formula on a relationship graph.

The latter point is crucial, since existing attempts to realize the aforementioned features appear to do so at the expense of losing efficieny of policy evaluation.

We argue here that a natural and well-established extension of modal logic – *hybrid logic* [5] – can overcome these shortcomings and provide a robust mathematical foundation for relationship-based access control.

A key contribution of this paper is the demonstration that hybrid logic has fragments well-suited to the needs of relationship-based access control, and that established results from the theory of hybrid logic are relevant and useful in their application to relationship-based access control.

$$
\begin{array}{rcl}
t & ::= & n \mid x \\
\phi & ::= & t \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle i\rangle\phi \mid \langle -i\rangle\phi \mid \\
 & & @_t\,\phi \mid \downarrow x\,\phi
\end{array}
$$

Figure 1: The syntax of a simple hybrid logic HL, where n ranges over *Nom*, x ranges over *Var*, p ranges over AP, and i ranges over *I*.

The other principal contribution of this paper is that our proposed hybrid logic eliminates a known exponential penalty incurred in expressing important policy idioms such as "at least two colleagues" in existing modal logics for relationship-based access control [10, 12].

As further evidence of the utility of hybrid logic, we also devise a fragment of this hybrid logic that gives rise only to policies whose access-control decisions depend only on the connectedness structure between the owner and the requester of a resource in a network graph.

Outline of paper.

We present a hybrid logic suitable for access control in Section 2. We explain, in Section 3, how this hybrid logic can be used for an access-control model based on relationships. Section 4 is devoted to example policies written in this hybrid logic, and to useful policy idioms. A local model-checking algorithm for policy decisions is given in Section 5. In Section 6, we map known fragments of hybrid logic into our context and study their expressiveness. In Section 7, a type system is developed for a fragment of our hybrid logic and it is shown that type-safe such formulas determine so-called *relational* policies. A discussion of issues not fully developed in this paper is found in Section 8, related work is reflected upon in Section 9, and the paper concludes in Section 10. The appendix, Section A, contains selected proofs at the discretion of reviewers.

2. HYBRID LOGIC

We define the syntax and semantics of hybrid logic, with a view towards its application in policy-based access control.

Syntax.

We take as given a set *Nom* of *nominal* symbols, an infinite set *Var* of *variable* symbols, a set I of *label* symbols, and a set AP of *propositional* symbols. Nominals and variables allow us to bind nodes in relationship graphs to principals. Labels represent the different relationships present in that graph. Using these sets of symbols, we define formulas of a hybrid logic HL in Fig. 1. Other logical symbols can be derived in the usual way. For example,

$$
\begin{array}{ll}
[i]\phi \stackrel{\text{def}}{=} \neg\langle i\rangle\neg\phi & \phi \vee \psi \stackrel{\text{def}}{=} \neg(\neg\phi \wedge \neg\psi) \\
\bot \stackrel{\text{def}}{=} p \wedge \neg p & \top \stackrel{\text{def}}{=} p \vee \neg p
\end{array}
$$

express a must modality for label i, disjunction, falsity, and truth (respectively).

Semantics.

We define models for HL, which express relationship graphs for access control, as well as valuations, which map variables to nodes in relationship graphs.

$$
\begin{array}{lcll}
M,s,g & \models & x & \stackrel{\text{def}}{=} \; s = g(x) \\
M,s,g & \models & n & \stackrel{\text{def}}{=} \; V(n) = \{s\} \\
M,s,g & \models & p & \stackrel{\text{def}}{=} \; s \in V(p) \\
M,s,g & \models & \neg\phi & \stackrel{\text{def}}{=} \; M,s,g \not\models \phi \\
M,s,g & \models & \phi_1 \wedge \phi_2 & \stackrel{\text{def}}{=} \; M,s,g \models \phi_1 \text{ and } M,s,g \models \phi_2 \\
M,s,g & \models & \langle i\rangle\phi & \stackrel{\text{def}}{=} \; M,s',g \models \phi \text{ for some } (s,s') \in R_i \\
M,s,g & \models & \langle -i\rangle\phi & \stackrel{\text{def}}{=} \; M,s',g \models \phi \text{ for some } (s',s) \in R_i \\
M,s,g & \models & @_n\,\phi & \stackrel{\text{def}}{=} \; M,s^*,g \models \phi \text{ where } V(n) = \{s^*\} \\
M,s,g & \models & @_x\,\phi & \stackrel{\text{def}}{=} \; M,g(x),g \models \phi \\
M,s,g & \models & \downarrow x\,\phi & \stackrel{\text{def}}{=} \; M,s,g[x \mapsto s] \models \phi
\end{array}
$$

Figure 2: Satisfaction relation $M,s,g \models \phi$ specifying that formula ϕ of hybrid logic HL is true in node s of model M under valuation g.

DEFINITION 1. *1. A* model M *of* HL *is a triple*

$$(S, \{R_i \subseteq S \times S \mid i \in I\}, V) \tag{1}$$

where S is a non-empty set of nodes, R_i a binary relation on S for all $i \in I$, and $V\colon (Nom \cup AP) \to 2^S$ a total function with $V(n)$ a singleton for all n in Nom.

2. A valuation $g\colon Var \to S$ *is a total function. For some such g, we write $g[x \mapsto s]$ for the valuation that maps x to s, and maps t to $g(t)$ if $t \neq x$.*

The intuition behind function V is that $V(p)$ is the set of nodes at which p is true, and that $V(n)$ is the set containing the unique node in the relationship graph of M "named" n by V. Valuations g are total functions from variables to nodes. Note that attributes of principals can be expressed in models of HL through propositions in set AP.

The meaning of formulas in HL is defined by a satisfaction relation $M,s,g \models \phi$, defined inductively in Figure 2.

Each nominal and variable is true at a single node. The meaning of nominals is specified through function V; the meaning of variables through valuation g. Propositions, on the other hand, are true at zero, one, or more nodes.

The meanings of conjunction and negation are standard, as is the meaning of the modalities. Formula $\langle i\rangle\phi$ holds at s if there is some R_i-successor s' of s such that ϕ holds at s'. Dually, $\langle -i\rangle\phi$ holds at s if there is some R_i-predeccesor $\tilde{s}$ of S such that ϕ holds at $\tilde{s}$.

We now discuss the *hybrid* operators of HL. Intuitively, $@_t\,\phi$ jumps to the node named by t, whereas $\downarrow x\,\phi$ binds the current node to variable x. Formally, formula $@_t\,\phi$ says that ϕ holds at the unique node identified by t with respect to function V (if t is a nominal) or valuation g (if t is a variable). Formula $\downarrow x\,\phi$ holds at node s if ϕ holds at s, but where now valuation g is updated so that x identifies s.

Notational conventions.

Operator $\downarrow x$ is a binding operator with the usual notions of free and bound variables. We fix some notation for satisfaction checks.

DEFINITION 2. *Let ϕ be a formula of* HL.

1. If ϕ contains no free variables, we write $M,s \models \phi$ for any $M,s,g \models \phi$, as the latter is independent of the choice of valuation g.

2. *If ϕ contains only free variables $x_1, \ldots, x_n$, we write $M, s, [x_1 \mapsto s_1, \ldots, x_n \mapsto s_n] \models \phi$ for any $M, s, g[x_1 \mapsto s_1, \ldots, x_n \mapsto s_n] \models \phi$, as the latter is again independent of the choice of g.*
3. *If ϕ is the Boolean combination of formulas of the form $@_t\,\psi$, we write $M, g \models \phi$, as then either $M, s, g \models \phi$ is true at all nodes s or false at all nodes s.*

We now use our hybrid logic HL to define an access-control model, focusing on the policy-decision point of such a model.

3. ACCESS-CONTROL MODEL

We outline here an archetypical model for relationship-based access control that uses hybrid logic for expressing access-control policies.

Protection state.

A protection state is simply a model M of HL in which the elements of S denote principals. The binary relations R_i represent interpersonal relations tracked by the access-control system. Each label i in I identifies one type of relationship (e.g., "parent"), such that (s_1, s_2) is in R_i iff principals s_1 and s_2 participate in a relationship of type i. In short, $(S, \{R_i \mid i \in I\})$ represents a directed, poly-relational social network. The valuation V specifies attributes of the principals. Each propositional symbol p in AP denotes an attribute. If a principal s in S has the attribute p, then s is in $V(p)$. Similarly, globally significant principals (e.g., trusted authorities) are identified by nominals via V.

Fragment of HL *for specifying access-control policies.*

Principals own resources and seek access on resources. One associates with each object *obj* a formula ϕ in HL that expresses a relationship between two parties: the *owner* and the *requester*. Then the requester is permitted access to *obj* if the owner and requester are in the relationship specified by ϕ. This approach has been suggested in [10] already, for a different logic, and it is what makes the access-control model *relationship-based*.

We propose to use two distinguished variables own and req to denote the principal who is the owner and the requester of the implicit object, respectively.

DEFINITION 3. *Let* $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ *be the set of formulas of* HL *that*

- *contain at most* own *and* req *as free variable, and*
- *are Boolean combinations of formulas of form* $@_{\mathsf{own}}\,\phi$ *and* $@_{\mathsf{req}}\,\psi$.

We refer to formulas of $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ as *policies*. A common policy pattern is a conjunction

$$@_{\mathsf{own}}\,\phi \wedge @_{\mathsf{req}}\,\psi \tag{2}$$

Sub-policy $@_{\mathsf{own}}\,\phi$ represents the owner's perspective, while $@_{\mathsf{req}}\,\psi$ represents the requester's perspective. Note that (2) is consistent with using only one such perspective. For example, dropping the second conjunct amounts to choosing ψ to be $\top$ and so the righthand conjunct is redundant.

Authorization decision.

The access control system arrives at an authorization decision as follows. Given a protection state (i.e., a model) M, a requesting principal r has permission to access an object *obj* in *Obj* that is controlled by principal o according to a policy *pol* in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ iff the following condition holds:

$$M, [\mathsf{own} \mapsto o, \mathsf{req} \mapsto r] \models \mathit{pol} \tag{3}$$

Intuitively, the only free variables own and req of formula *pol* of $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ get bound to the principals named by the owner and requester, respectively, and this formula is then evaluated with those bindings.

Thus, checking whether an access is granted amounts to checking the satisfaction of a $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ formula with respect to a valuation. We discuss a local model-checking algorithm for $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ below.

Policy functions.

Through the semantics of hybrid logic, which has been defined as a satisfaction relation, every policy ϕ in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ induces a *policy function*, which maps protection states to binary *permission relations*.

DEFINITION 4. *1. A policy function P is a total function $M = (S_M, \ldots) \mapsto P(M)$ from models M to binary relations $P(M) \subseteq S_M \times S_M$.*
2. The policy function induced by a policy ϕ of $\mathsf{HL}(\mathsf{own}, \mathsf{req})$*, written* $\mathsf{p}[\phi]$*, is defined as follows:*

$$\mathsf{p}[\phi] = \{(o, r) \in S_M \times S_M \mid M, [\mathsf{own} \mapsto o, \mathsf{req} \mapsto r] \models \phi\}$$

The intuition behind policy functions is that a requester r can access an object of owner o in protection state M if (o, r) is in $P(M)$ – and is denied access if (o, r) is not in $P(M)$. Note that a policy function maps a collection of relations (plus a mapping for nominals and propositions) over a set of principals to a single permission relation over the same set.

Below we will investigate what sort of policy functions are induced by formulas in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$.

Discussion.

The above access-control model can easily be generalized. For example, the OASIS standard XACML allows for multiple subjects [16] (i.e., requesters) in a single access. In hybrid logic, we could use such an idea to model, for example, a threshold scheme for access so that multiple requesters are required, with specified relationships in place; e.g.

$$M, g \models @_{\mathsf{own}}\,(\langle \mathit{friend} \rangle \mathsf{req}_1 \wedge \langle \mathit{parent} \rangle \mathsf{req}_2) \wedge @_{\mathsf{req}_1}\,\neg \mathit{Bob} \tag{4}$$

might model an access request where a permission requires two requesters: a friend who is not Bob, and a parent.

We mention these possibilities as expressiveness results should ideally not depend on how the logic might interface with the access-control model. That is to say, our implementation of a policy-decision point should readily support such extensions without a need for major modifications. We will revisit such issues below and will also see that our approach does indeed accommodate this flexibility.

4. EXAMPLE POLICIES

We now provide example policies, starting with basic ones.

Basic policies.

If the owner of an object defines the policy to be

$$@_{\mathsf{own}}\,\langle \mathit{friend} \rangle \mathsf{req} \tag{5}$$

then every friend of the resource owner is granted access to that resource. Note that this policy specifies no constraints from the perspective of the requester. Similarly, the formula

$$@_{\mathsf{own}}\,\langle friend\rangle(\mathsf{req} \vee \langle friend\rangle\mathsf{req}) \tag{6}$$

captures a "friend or a friend of a friend" policy. The formula

$$@_{\mathsf{own}}\,(\langle teammate\rangle\mathsf{req} \wedge \langle friend\rangle\mathsf{req}) \tag{7}$$

specifies that all friends who are teammates get access. This example already hints at the usefulness of hybrid logic. The use of the variable req allows us to refer to some principal who is *both* a teammate and a friend of the owner.

If req behaved like a normal proposition that could be true at zero or more nodes, then we could not capture this semantic conjunction in the logic; the existential quantification of $\langle i\rangle$ does not distribute through conjunction.

In a similar vein we can specify formulas

$$@_{\mathsf{own}}\,\langle parent\rangle\langle parent\rangle\mathsf{req} \tag{8}$$

$$@_{\mathsf{own}}\,\langle sibling\rangle(\mathsf{req} \wedge [spouse]\bot) \tag{9}$$

$$@_{\mathsf{own}}\,\langle child\rangle\mathsf{req} \wedge [child]\mathsf{req} \tag{10}$$

which express the respective policies "grant access to grandparents", "grant access to unmarried siblings", and "grant access if requester is sole child of the owner" (respectively).

The policies we have specified so far are variants or adaptations of policies written the logic E of [12]. To see the true benefits of using HL, we write more complex policies next.

More complex policies.

Consider a teacher who wants to confine access to her picture to friends who are teachers, but who also wants to grant access to only those friends of such friends who are not taught by friends. This is hard to express in English, but there is a true case of an English teacher who had to resign because her picture could be seen by student friends of friends – unbeknownst to her. Hybrid logic can express such a policy unambigiously. One possible formula is

$$@_{\mathsf{own}}\,(\langle friend\rangle(\mathsf{req} \wedge isTeacher) \vee \langle friend\rangle(isTeacher \wedge \langle friend\rangle\mathsf{req} \wedge \neg\langle student\rangle\mathsf{req})) \tag{11}$$

Note that this formula in indeed in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$. The first disjunct specifies that access is granted if the requester is a friend who is a teacher, where $isTeacher$ is in AP. The second disjunct specifies that access is granted if the requester is a friend of a teacher friend of the owner, but where the requester is also not a student of that teacher friend.

We point out that this policy does not prevent the scenario where access is granted to *some* student of a teacher friend, since the may modality is an existential quantification.

This policy also illustrates the utility of propositions (attributes), as we here want to express that a friend is *some* teacher, not necessarily the teacher or student of the owner or requester. This property could be expressed through $\langle teacher\rangle\top$, someone is a teacher if they teach someone. But such an encoding becomes ackward for unary relations such as $isDiabetic$, so we do include propositions in our logic HL.

Dual policies.

The last example illustrates that we can compose negative and positive permissions. For example, the policy

$$@_{\mathsf{own}}\,\langle friend\rangle(\mathsf{req} \wedge \neg Alice) \tag{12}$$

says access is granted to all friends, except to *Alice*, a principal that is a nominal in *Nom*. Note that this policy makes no assumption about whether or not *Alice* is actually a friend.

The policy expressed in (12) serves also as an example for how one might "dualize" a policy written from the perspective of one of the principals into an equivalent one written from the perspective of the other principal. The policy

$$@_{\mathsf{req}}\,(\langle -friend\rangle\mathsf{own} \wedge \neg Alice) \tag{13}$$

is intuitively equivalent to that in (12), but of form $@_{\mathsf{req}}\,\phi$.

Graded modalities.

Another nice feature of hybrid logic is that it allows us to express so called *graded* may modalities [5] as syntactic sugar of the logic. To see this in HL, let n be a positive natural number, and let $\langle i\rangle_n$ be the corresponding graded may modality for label i. The intuition of $\langle i\rangle_n\phi$ is then that it holds in node s iff there are *at least* n many R_i-successors of s at which ϕ holds.

For example, a policy that grants access if there are at least 3 friends of the owner who satisfy ϕ can be specified as

$$@_{\mathsf{own}}\,\langle friend\rangle_3\phi \tag{14}$$

To see that $\langle i\rangle_n\phi$ is expressible in HL, take $n+1$ many variables x, y_1, y_2, $\ldots$ y_n that do not occur in ϕ and define

$$\begin{aligned}\langle i\rangle_n\phi \;\stackrel{\text{def}}{=}\; & \downarrow x\,\langle i\rangle\downarrow y_1\,(\phi \wedge \\ & @_x\,\langle i\rangle\downarrow y_2\,(\neg y_1 \wedge \phi \wedge \\ & @_x\,\langle i\rangle\downarrow y_3\,(\neg y_1 \wedge \neg y_2 \wedge \phi \wedge \\ & \cdots \\ & @_x\,\langle i\rangle\downarrow y_n\,(\neg y_1 \wedge \neg y_2 \wedge \cdots \wedge \neg y_{n-1} \wedge \phi))\ldots)\end{aligned} \tag{15}$$

This standard encoding says that at the current node, named x, there is some R_i-successor y_1 of x that satisfies ϕ, and there is some R_i-successor y_2 of x that is different from y_1 and satisfies ϕ, and so on. This clearly renders the desired semantic effect of having at least n many R_i-successors of the currrent node that satisfy ϕ.

This encoding is complex, but that complexity is of no genuine concern to policy specifiers, as they would only see and use the syntactic sugar $\langle i\rangle_n$.

Graded modalities give us the ability to count exactly as well. To specify that there are exactly n R_i-successors that satisfy ϕ, we may write

$$\langle i\rangle_{=n}\phi \stackrel{\text{def}}{=} \langle i\rangle_n\phi \wedge \neg\langle i\rangle_{n+1}\phi \tag{16}$$

One can model a rudimentary trust level in a network of friends by asking whether the requester is connected to the owner by a path of *friend* labels of length at most k.

We can specify such policies inductively for $k \geq 1$ as

$$\begin{aligned}\mathsf{depth}[friend, 1] &\stackrel{\text{def}}{=} \langle friend\rangle\mathsf{req} \\ \mathsf{depth}[friend, k+1] &\stackrel{\text{def}}{=} \mathsf{depth}[friend, k] \vee \langle friend\rangle\mathsf{depth}[friend, k]\end{aligned} \tag{17}$$

We can combine this trust mechanism with graded modalities to express a policy that grants access if there are at least two colleagues who have sufficient trust in the requester:

$$@_{\mathsf{own}}\,\langle colleague\rangle_2\mathsf{depth}[friend, 3] \tag{18}$$

Next, let us now consider a policy cf_k based on common friends. It grants access to the owner, to his friends, and to

those who have at least $k > 0$ common friends. Through graded may modality, we can express this in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ as

$$\mathsf{cf}_k \stackrel{\text{def}}{=} @_{\mathsf{own}}\,(\mathsf{req} \vee \langle friend\rangle \mathsf{req} \vee \langle friend\rangle_k \langle friend\rangle \mathsf{req}) \quad (19)$$

The leftmost disjunct specifies that the owner has access, the second disjunct says that friends of the owner have access, and the third disjunct says that access is granted to requesters who have at least k many friends in common with the owner. The encoding assumes that R_{friend} is symmetric. If not, the last modality should be an inverse one.

Here is an example policy of $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ that has non-dual subpolicies from the perspective of owner and requester:

$$@_{\mathsf{own}}\,(\langle friend\rangle \mathsf{req} \wedge \langle friend\rangle_3 \top) \wedge @_{\mathsf{req}}\,\langle friend\rangle_5 \neg \mathsf{own} \quad (20)$$

This policy grants access if the requester is a friend of the owner, the owner has at least three friends (counting or not counting the requester), and if additionally the requester has at least five friends other than the owner. Intuitively, in order to express this policy in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ we seem to require both operators $@_{\mathsf{req}}$ and $@_{\mathsf{own}}$ as the policy involves a form of counting in both nodes named by own and req.

5. LOCAL MODEL CHECKING

Given a model M with node set S, a valuation g, and a formula ϕ of HL, we can compute the set of all nodes s in S for which $M, s, g \models \phi$ holds. This is known as *global* model checking. For hybrid logics, such global model checking algorithms have been developed [13]. For HL, these algorithms are linear in the size of the model and the formula.

But global model checking is often ill suited for our access-control model, as its protection state may be huge whereas only a small portion of it may be needed to make an access-control decision. Linear complexity, e.g., will not help if one wants to evaluate the entire Facebook relationship graph.

In our setting, we are given a model M as protection state, and a policy *pol* in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ as specification of access control. We then wish to decide whether $M, [\mathsf{own} \mapsto o, \mathsf{req} \mapsto r] \models pol$ for specified nodes o and r.

This form of evaluation is a kind of *local* model checking. The term "local" is used because the intuition is that the model M is explored only as needed from the nodes of interest, those named by own and req.

Local model checking is a better fit for our needs as only those portions of the model that are potentially needed to make an access-control decision are explored.

Description of algorithm.

We now describe our local model-checking algorithm for $\mathsf{HL}(\mathsf{own}, \mathsf{req})$. Its pseudo-code is depicted in Figure 3. The model M is implicit but its structure and local nodes and valuations are explicit in the code.

The algorithm `MC` has as argument a local node s, a valuation g, and a formula ϕ of HL. Its body is a case analysis of the top-level operator of ϕ. We assume that $V(n)$ and $V(p)$ are implemented as lists, that `isElem` is a membership test for such lists, that `hd` returns the first element of a non-empty list, and that `areEqual` can check for node equality.

The algorithm does a recursive decent until it encounters formulas that are variables, nominals, or propositions. The cases of the forward and backward modalities require a call to a sub-routine `MCmay`.

```
MC(s,g,φ) {
  case {
    φ is x : return (areEqual(s,g(x)));
    φ is n : return (isElem(s,V(n)));
    φ is p : return (isElem(s,V(p)));
    φ is ¬ψ : return (!MC(s,g,ψ));
    φ is ψ1 ∧ ψ2 : return (MC(s,g,ψ1) && MC(s,g,ψ2));
    φ is ⟨i⟩ψ : return MCmay(s,g,ψ,i,fwd);
    φ is ⟨−i⟩ψ : return MCmay(s,g,ψ,i,bwd);
    φ is @n ψ : let t = hd(V(n)) in return MC(t,g,ψ);
    φ is @x ψ : let t = g(x) in return MC(t,g,ψ);
    φ is ↓x ψ : let h = g[x ↦ s] in return MC(s,h,ψ);
   }
}

MCmay(s,g,φ,i,direction) {
  if (direction == fwd) { X = [ s′ | (s,s′) in Ri ];
  } else                { X = [ s′ | (s′,s) in Ri ]; }
  for (all s′ in X) {
      if (MC(s′,g,φ)) { return true; }
   }
   return false;
}

policyDecision(o,r,φ) {  // φ in HL(own, req)
  let g = [own ↦ o, req ↦ r] in {
    return MC(o,g,φ);
  }
}
```

Figure 3: Local model checking algorithm for HL.

Routine `MCmay` first computes the set of all R_i-successors or predecessors of node s, where that choice is decided by an inspection of a parameter value that indicates whether the modality is a forward one $\langle i\rangle$ or a backward one $\langle -i\rangle$.

It then iterates through all these nodes until one of them makes the formula true, in which case it does return `true`. If no such node makes the formula true, it returns `false`.

A genuine implementation of algorithm `MC` would *not* precompute the sets X, but would generate new elements of X on demand until either a witness for truth has been found, or all elements of X are revealed not to be such witnesses. Also, to make this efficient each reached node would keep a hash table of subformulas it has already evaluated so that each subformula is evaluated at most once in each node.

We can use that local model-checking algorithm to implement a policy decision point in `policyDecision`. It takes as input a node o representing the owner of the object, a node r representing the requester of that object, and a formula ϕ of $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ representing the access-control policy. Then it creates the valuation that binds own and req to o and r, respectively. Finally, it returns the result of the local model check on ϕ under valuation g.

Note that the argument for the node supplied to this local model check does not matter as the evaluation of a formula of $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ is independent of a given node (although it will be evaluated at local nodes as demanded by occurrences of $@_t$ and $\downarrow x$ within the formula). We simply named that object o as algorithm `MC` requires a value for this parameter.

Beyond local model checking.

As already discussed, the local approach to model checking seems preferrable to the global one.

But our algorithm `policyDecision` does not really exploit that the formula to be checked is from $\mathsf{HL}(\mathsf{own}, \mathsf{req})$: it delegates the check to the algorithm `MC`, which "forgets" this fact

$$a ::= n \mid \mathsf{own} \mid \mathsf{req}$$
$$\phi ::= a \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle i\rangle\phi \mid \langle -i\rangle\phi \mid @_a\,\phi$$

Figure 4: The syntax of the fragment $\mathsf{HL}(@, \mathsf{own}, \mathsf{req})$ of $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ that bans operator $\downarrow x$.

and assumes that the formula is any from HL.

It would therefore be of interest to determine whether knowledge that a formula is in $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ can be exploited in model checking. One idea is to combine a forward search from own with a backwards search from req so that only relevant portions of the overall set of nodes are ever explored.

6. POLICY EXPRESSIVENESS

Our proposal to use hybrid logic for the specification and enforcement of relationship-based access control warrants looking into the kinds of properties that policies written in HL enjoy, especially properties that shed light on the expressiveness of fragments of HL.

Binder-free policies and hybrid bisimulation.

Figure 4 gives the definition of the fragment $\mathsf{HL}(@, \mathsf{own}, \mathsf{req})$ of HL. This fragment consists of policies not containing the binding operator $\downarrow x$. We call such policies *binder-free policies*. Policies of $\mathsf{HL}(@, \mathsf{own}, \mathsf{req})$ do not contain the graded modalities, as their encoding relies on additional variables x and the operator $\downarrow x$. However, all our policy examples so far that do not use graded modalities were written in $\mathsf{HL}(@, \mathsf{own}, \mathsf{req})$.

So what sort of policies can actually be expressed in the fragment $\mathsf{HL}(@, \mathsf{own}, \mathsf{req})$? To understand this better, we use an equivalence relation on models that was adaped from modal logic. We will ask whether a policy will make the same access decisions on any two equivalent models.

Definition 5. *Let $M = (S_M, \{R_i^M \subseteq S_M \times S_M \mid i \in I\}, V_M)$ and $N = (S_N, \{R_i^N \subseteq S_N \times S_N \mid i \in I\}, V_N)$ be models of hybrid logic with respective valuations g and h.*

1. *A relation $\rho \subseteq S_M \times S_N$ is a* forward hybrid bisimulation *between M and N iff*
   - **nom** *for all n in Nom, $(V_M(n), V_N(n))$ is in ρ*
   - **forth** *if (s,t) is in ρ, then for all i and (s,s') in R_i^M, there is some (t,t') in R_i^N where (s',t') is in ρ*
   - **back** *if (s,t) is in ρ, then for all i and (t,t') in R_i^N, there is some (s,s') in R_i^M where (s',t') is in ρ*
2. *Such a relation ρ is a* hybrid bisimulation *between M and N iff it additionally satisfies*
   - **-forth** *if (s,t) is in ρ, then for all i and (s,s') in R_{-i}^M, there is some (t,t') in R_{-i}^N where (s',t') is in ρ*
   - **-back** *if (s,t) is in ρ, then for all i and (t,t') in R_{-i}^N, there is some (s,s') in R_{-i}^M where (s',t') is in ρ*
3. *A pair of nodes (s,t) is bisimilar (eliding adjective "hybrid") iff (s,t) is in ρ for some hybrid bisimulation ρ.*
4. *We write $M, s, g \sim_\rho^X N, t, h$ if ρ is a hybrid bisimulation over M and N with $(s,t) \in \rho$ and $(g(x), h(x)) \in \rho$ for all x in variable set X.*
5. *We say that M, g and N, h are bisimilar for a set of variables X iff $M, s, g \sim_\rho^X N, t, h$ for some ρ.*

Forward bisimulations allow only forward moves in the graph, whereas bisimulations also allow backward moves.

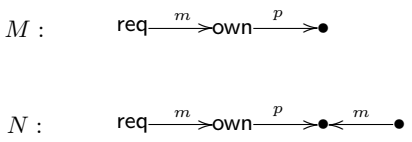

Figure 5: Two models M and N bisimilar in the forward sense, but not in the backwards sense.

In Figure 5 we see models M and N and valuations g and h such that M, g and N, h are forward bisimilar for the set of variables $\{\mathsf{own}, \mathsf{req}\}$. Model N has an additional node over M, but this does not affect forward bisimulation, as its additional node is unreachable from any variable (req or own) or nominal (there are none here).

However, if we use full bisimulation, M, g and N, h are no longer bisimilar for $\{\mathsf{own}, \mathsf{req}\}$. For, assume, by way of contradiction, that (s,t), defined as $(g(\mathsf{own}), h(\mathsf{own}))$, is in a hybrid bisimulation. By condition **-forth**, we have (s,s') in R_{-m}^M where s' is the one such m-predecessor not named by req. Bisimulation therefore would require some t' in N such that (t,t') is in R_{-m}^N and where (s',t') are bisimilar. But this is impossible, as (t,t') in R_{-m}^N implies that t' is named by req. But s' is not named by req, a contradiction.

We can use bisimilarity to help understand the expressiveness of $\mathsf{HL}(@)$, the fragment of HL without the binder $\downarrow x$. Clearly, $\mathsf{HL}(@)$ contains $\mathsf{HL}(@, \mathsf{own}, \mathsf{req})$. Do bisimilar models satisfy the same formulas of $\mathsf{HL}(@)$? If so, only features of models that are "observed" by bisimilarity are expressible by policies of $\mathsf{HL}(@)$. We now define this idea precisely.

Definition 6. *Let X be the set of free variables in a formula ϕ of* HL. *Then ϕ is* closed under bisimulation *if, for all models M, N, nodes s, t of M and N, and valuations g and h*

$$M, s, g \sim_\rho^X N, t, h \text{ implies } (M, s, g \models \phi \text{ iff } M, t, h \models \phi).$$

A fragment of HL *is closed under bisimulation if every formula in the fragment is.*

A standard result is that $\mathsf{HL}(@)$ *is* closed under bisimulation. In fact, more is known: if a pair of nodes is not bisimilar, then there is some formula of $\mathsf{HL}(@)$ that holds in one of these nodes and not the other. To connect this to our example, such a formula is $@_{\mathsf{own}}\,[-m]\mathsf{req}$, as M satisfies it but N does not. Because $\mathsf{HL}(@)$ is closed under bisimulation, and $\mathsf{HL}(@, \mathsf{own}, \mathsf{req})$ is a fragment of $\mathsf{HL}(@)$, policies of $\mathsf{HL}(@, \mathsf{own}, \mathsf{req})$ are also closed under bisimulation.

Let us move from closing policies under bisimulation to closing policy functions under bisimulation. We shall write $M, (o_M, r_M) \sim_\rho N, (o_N, r_N)$ if ρ is a hybrid bisimulation between M and N such that $(o_M, o_N) \in \rho$ and $(r_M, r_N) \in \rho$.

Definition 7. *A policy function P is* closed under bisimulation *if, for all models M and N*

$$M, (o_M, r_M) \sim_\rho N, (o_N, r_N) \text{ implies } P(M)(o_M, r_M) \text{ iff } P(N)(o_N, r_N).$$

Since formulas of $\mathsf{HL}(@, \mathsf{own}, \mathsf{req})$ are closed under hybrid bisimulation, the policy functions induced by such formulas are as well.

Proposition 1. *Binder-free policies are closed under hybrid bisimulation.*

It is natural to ask whether a converse of this fact also holds – can every policy function closed under hybrid bisimulation be expressed by a binder-free policy? We cannot expect this in general. Indeed, the policy function may not even be expressible in first-order logic (of which HL and so also HL(@, own, req) are essentially a fragment).

However, every first-order logic formula closed under hybrid bisimulation is semantically equivalent to a formula of hybrid logic HL(@) (see, e.g., Theorem 4.14 in [3] and Theorem 14 in [4]).

Policies and their generated submodels.

It is well known that HL corresponds syntactically to the bounded fragment of first-order logic (see, e.g., Corollary 6.4 in [3] and Theorem 18 in [4]). The latter restricts existential quantifications to the form $\exists x\colon (R_i(s,x) \wedge \cdot)$, and universal quantification to the form $\forall x\colon (R_i(s,x) \rightarrow \cdot)$ where s is any term distinct from x.

There is also a semantic characterization of this fragment of first-order logic, and therefore of HL. This involves the notion of generated submodel. Intuitively, a generated submodel is a model that is reduced by eliminating elements of the model not relevant to the variables in a given set.

DEFINITION 8. *Let $M = (S, \{R_i \mid i \in I\}, V)$ be a model with set of nominals Nom. Let X be a set of variables and g a valuation for M. The submodel of M, g generated by X, denoted by $\langle M, g\rangle_X$*

- *has as set of nodes S^X those nodes that are reachable from set $\{V(n) \mid n \in Nom\} \cup \{g(x) \mid x \in X\}$ via relation $\bigcup_{i\in I}(R_i \cup R_{-i})$*
- *as total function V^X, the restriction of V to set S^X*
- *as binary relation R_i^X the restriction $R_i \cap (S^X \times S^X)$ of R_i onto S^X.*

In our setting, formulas of HL(own, req) have at most req and own as free variables, so X will be a subset of {own, req}. A standard result then implies that such formulas are invariant under submodels generated by {own, req}.

THEOREM 1. *All formulas ϕ of* HL(own, req) *are invariant under submodels generated by* {own, req}*, i.e., for all models M and valuations g, we have*

$$M, g \models \phi \qquad \textit{iff} \qquad \langle M, g\rangle_X, g \models \phi \tag{21}$$

Consider the model M in Figure 6. The generated submodel $\langle M, g\rangle_X$ for X = {own, req} is obtained from M by removing the 3-clique of "anonymous" friends. We cannot remove nodes that are named by own, req, or any nominal in M (here *Alice*). Therefore, all relationships between these nodes are preserved. The 3-clique disappears as none of its nodes is reachable from a named node.

If M were to be modified so that one of the 3-clique nodes had a nominal, say *Bob*, as name, then $\langle M, g\rangle_X$ would equal M, as then the other two nodes in that 3-clique would also be reachable from a named node.

It is of interest to determine whether one can benefit from this invariance result in the derivation of improved local model-checking algorithms. In particular, these benefits may turn out to be considerable for formulas written in the logic HL(own, req). We leave the investigation of this for future work.

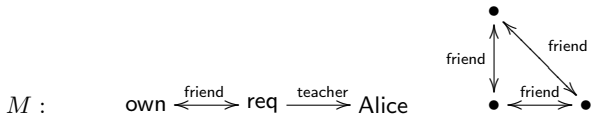

Figure 6: A model M whose valuation g maps own and req to the indicated nodes.

7. RELATIONAL POLICIES

Previous work [2, 12] has noted that what distinguishes relationship-based access control from traditional access-control paradigms is its extensive use of *relational policies*. Intuitively, a relational policy is one in which the authorization decision is based *solely* on how the owner and the requester are connected to one another (e.g., friend, friend-of-friend, etc). Therefore, a relational policy does not base its authorization decision on the requester's identity (e.g., John can access), attributes (e.g., managers can access), or social positions (e.g., those who have at least 100 friends can access).

The goal of this section is to identify a syntactic fragment of the logic HL(own, req) that captures relational policies. This then allows policy developers to efficiently verify that their policy specifications determine relational policies.

Relational policies.

We begin by formalizing what it means for a policy function to be relational. We first define some auxiliary concepts that use only the binary relations R_i of models.

DEFINITION 9. *Let M and N be models.*

1. *These models are* isomorphic via *a bijection $f\colon S_M \rightarrow S_N$, if, for all $i \in I\colon (s,t) \in R_i^M$ iff $(f(s), f(t)) \in R_i^N$.*
2. *Nodes s, t of S_M are* connected, *written $s \overset{M}{\rightsquigarrow} t$, if either $s = t$, or inductively, there is an s' in S_M with $(s, s') \in R_i^M \cup (R_i^M)^{-1}$, for some $i \in I$, and $s' \overset{M}{\rightsquigarrow} t$.*
3. *The* shared component *of nodes s, t in M, written $\mathsf{SC}(M, s, t)$, is the relational structure $(S, \{R_i \subseteq S \times S \mid i \in I\})$ defined as follows:*

$$S \overset{\text{def}}{=} \{s,t\} \cup \{s' \in S_M \mid s \overset{M}{\rightsquigarrow} s' \ \wedge \ s' \overset{M}{\rightsquigarrow} t\}$$

$$R_i \overset{\text{def}}{=} R_i^M \cap (S \times S)$$

There are two parts to what it means to be a relational policy function. First, the policy function must be "topology-based": it must consume neither attribute information (i.e., valuations are not considered) nor "identity information" (i.e., isomorphic labeled graphs cannot be distinguished) when an authorization decision is made.

DEFINITION 10. *A policy function P is* topology-based *if, for all models M and N, and all bijections $f : S_M \rightarrow S_N$, whenever M and N are isomorphic via f, then $P(N) = \{(f(s), f(t) \mid (s,t) \in P(M)\}$.*

Second, the policy function must be "local": changes in permission to a model must reflect a change in connectivity between the owner and requester.

DEFINITION 11. *A policy function P is* local *iff, for all models M and N, and all $s, t \in S_M \cap S_N$, if*

$$\mathsf{SC}(M, s, t) = \mathsf{SC}(N, s, t) \textit{ implies } P(M)(s,t) = P(N)(s,t).$$

$$\begin{array}{rcl} \psi & ::= & \top \mid \bot \mid x \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \\ & & \langle i\rangle\psi \mid \langle -i\rangle\psi \mid [i]\psi \mid [-i]\psi \mid @_x\,\psi \mid \downarrow x\,\psi \\ \phi & ::= & \top \mid \bot \mid @_{\mathsf{own}}\,\psi \mid @_{\mathsf{req}}\,\psi \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \end{array}$$

Figure 7: Syntax of candidate formulas for a relational fragment of $\mathsf{HL}(\mathsf{own},\mathsf{req})$, with x from Var.

The definition of "local" demands that any change in an access decision must imply that either (a) s and t have gone from being disconnected to being connected (or vice versa), or (b) the shared component of s and t have been altered.

Such a requirement ensures that any change of authorization decision is not caused merely by the social positions of the requester or the owner (i.e., where exactly they are in the relational structure).

As Proposition 3 in Appendix A shows, this definition of local-ness is equivalent to the one given in [12].

Example. The following formulas express local policies:

$$@_{\mathsf{own}}\,(\langle\mathsf{child}\rangle\mathsf{req} \wedge [\mathsf{child}]\mathsf{req}) \tag{22}$$

$$@_{\mathsf{own}}\,\langle\mathsf{friend}\rangle(\mathsf{req} \wedge \langle\mathsf{spouse}\rangle\top) \tag{23}$$

Formula (22) demands the requester to be the *only* child of the owner. Formula (23) requires the requester to be a *married* friend of the owner.

The following formulas express policies that are not local:

$$@_{\mathsf{req}}\,\langle\mathsf{spouse}\rangle\top \tag{24}$$

$$@_{\mathsf{own}}\,[\mathsf{child}]\mathsf{req} \tag{25}$$

Formula (24), which is a relaxation of (23), requires the requester to be married. It is not local because a change of authorization decision only requires the introduction (or elimination) of a spouse edge in the neighbourhood of the requester, which may otherwise be disconnected from the owner.

Formula (25), which is a relaxation of (22), grants access either if the owner has no child, or if the requester is the only child. The following explains why it is not a local policy. Suppose we start with a model M in which the owner has no child, and the requester is disconnected from the owner. The requester has access in M. Now the owner gives birth to a child, resulting in a new model N, in which the owner is incident to a child edge. The requester loses its access in protection state N, but the owner and the requester remain disconnected. **(End of Example.)**

We can now define the technical notion of relational policy.

Definition 12. *A policy function is* relational *iff it is both topology-based and local.*

A relational fragment of $\mathsf{HL}(\mathsf{own},\mathsf{req})$.

It is easy to show that every formula in $\mathsf{HL}(\mathsf{own},\mathsf{req})$ expresses a topology-based policy. To obtain relational policies, the challenge is to ensure that formulas are constructed to express *only* local policies. To better appreciate the nature of this challenge, observe the following:

Proposition 2. *The family of local policies contains* $\mathsf{p}[\top]$ *and* $\mathsf{p}[\bot]$ *and is closed under boolean combinations.*

Thus complex local policies can be built up from primitive ones using the boolean connectives offered by $\mathsf{HL}(\mathsf{own},\mathsf{req})$. It is the modal operators $\langle i\rangle$ and $\langle -i\rangle$ that may break local-ness of policies. The reason is that, in general, local-ness is not preserved by relational composition. Suppose P_1 and P_2 are policy functions. We write $P_1 \circ P_2$ to denote the policy function P such that $P(M) = \{(s,t) \in S_M \times S_M \mid \exists s' \in S_M\colon (s,s') \in P_1(M), (s',t) \in P_2(M)\}$. Even though policy functions P_1 and P_2 may both be local, $P_1 \circ P_2$ need not be local. An example is when P_2 is $\mathsf{p}[\top]$. Thus $\langle i\rangle\psi$ may not be local even if ψ is local. This is illustrated by formula (24).

We define a relational fragment of $\mathsf{HL}(\mathsf{own},\mathsf{req})$ to ensure that modal operators preserve local-ness. This relational fragment is obtained in two steps. First, we constrain the syntax of $\mathsf{HL}(\mathsf{own},\mathsf{req})$ by the grammar in Figure 7. Second, we impose a type system (Figure 8) to further constrain formula construction, such that only properly typed formulas belong to the fragment.

Definition 13. *The hybrid logic fragment* HL_{rel} *contains those formulas* ϕ *as defined in Figure 7 such that:*

- *For all subformulas* $@_{\mathsf{own}}\,\psi$ *of* ϕ*, we have* $\psi : \mathsf{Local}(\mathsf{req})$.
- *For all subformulas* $@_{\mathsf{req}}\,\psi$ *of* ϕ*, we have* $\psi : \mathsf{Local}(\mathsf{own})$.

We now explain the two steps of restricting $\mathsf{HL}(\mathsf{own},\mathsf{req})$ in turn. First, the syntax of ϕ in Figure 7 reiterates the requirement of $\mathsf{HL}(\mathsf{own},\mathsf{req})$, that policy formulas are boolean combinations of subformulas of the form "$@_{\mathsf{own}}\,\psi$" or "$@_{\mathsf{req}}\,\psi$". As long as these subformulas express local policies, then Proposition 2 ensures that ϕ is local.

Second, the inference rules in Figure 8 involve two type judgments of form "$\psi : \mathsf{Local}(x)$" and "$\psi : \mathsf{OC}(x)$", where x is typically either own or req. Intuitively, the derivation of judgment $\psi : \mathsf{Local}(\mathsf{own})$ means ψ can be used within $@_{\mathsf{own}}$ to form a local policy. The interpretation of $\mathsf{Local}(\mathsf{req})$ is similar. The judgment based on label $\mathsf{OC}(x)$ is for typing subformulas ψ that are "*owner-checkable (OC)*" as defined in [12]. Although OC formulas are not local, they can be combined with local formulas in a conjuction to yield local formulas. The definition of OC policies and its use in our proof of the next theorem can be found in the appendix.

Theorem 2 (Type Soundness). *If* $\psi : \mathsf{Local}(\mathsf{req})$*, then* $\mathsf{p}[@_{\mathsf{own}}\,\psi]$ *is a local policy. Similarly, if* $\psi : \mathsf{Local}(\mathsf{own})$*, then* $\mathsf{p}[@_{\mathsf{req}}\,\psi]$ *is a local policy.*

To see an application of this theorem, consider the policy in (23) of form $@_{\mathsf{own}}\,\psi$. The first conjunct type checks as $\mathsf{Local}(\mathsf{own})$, the second one as $\mathsf{OC}(\mathsf{own})$. So their conjunction and also ψ both type check as $\mathsf{Local}(\mathsf{own})$. Theorem 2 ensures the policy itself is local. The policy in (22) can be type checked in a similar fashion.

An attempt to type check the policy of form $@_{\mathsf{req}}\,\psi$ in (24), however, fails: our type system can assign type $\mathsf{OC}(\mathsf{req})$ to ψ but this cannot be lifted to type $\mathsf{Local}(\mathsf{req})$. Policy (25) fails to type check to $\mathsf{Local}(\mathsf{own})$ for a similar reason.

The encoding of graded modalities $\langle i\rangle_n\phi$ given in (15) will type check to $\mathsf{Local}(x)$ so long as ϕ type checks to $\mathsf{Local}(x)$.

Our typing rules can also type check the depth-first search tree encoding of finitary relational policies [12, Appendix A].

In summary, we have identified a relational fragment HL_{rel} via a grammar and a type system. This allows policy developers to efficiently verify whether the policy formula under development is indeed relational.

$$\frac{\psi : \mathsf{Local}(x)}{\psi : \mathsf{OC}(x)} \quad \text{(O-Sub)}$$

$$\frac{}{\top : \mathsf{OC}(x)} \quad \text{(O-Top)}$$

$$\frac{}{\bot : \mathsf{OC}(x)} \quad \text{(O-Bot)}$$

$$\frac{}{y : \mathsf{OC}(x)} \quad \text{(O-Var)}$$

$$\frac{\psi : \mathsf{OC}(x)}{\neg\psi : \mathsf{OC}(x)} \quad \text{(O-Not)}$$

$$\frac{\psi_1 : \mathsf{OC}(x) \qquad \psi_2 : \mathsf{OC}(x)}{\psi_1 \vee \psi_2 : \mathsf{OC}(x)} \quad \text{(O-Or)}$$

$$\frac{\psi_1 : \mathsf{OC}(x) \qquad \psi_2 : \mathsf{OC}(x)}{\psi_1 \wedge \psi_2 : \mathsf{OC}(x)} \quad \text{(O-And)}$$

$$\frac{\psi : \mathsf{OC}(x)}{\langle i\rangle\psi : \mathsf{OC}(x)} \quad \text{(O-May)}$$

$$\frac{\psi : \mathsf{OC}(x)}{[i]\psi : \mathsf{OC}(x)} \quad \text{(O-Mus)}$$

$$\frac{\psi : \mathsf{OC}(x) \qquad y \neq x}{@_y\,\psi : \mathsf{OC}(x)} \quad \text{(O-At)}$$

$$\frac{\psi : \mathsf{OC}(x) \qquad y \neq x}{\downarrow y\,\psi : \mathsf{OC}(x)} \quad \text{(O-Dow)}$$

$$\frac{}{\bot : \mathsf{Local}(x)} \quad \text{(L-Bot)}$$

$$\frac{}{x : \mathsf{Local}(x)} \quad \text{(L-Var)}$$

$$\frac{\psi_1 : \mathsf{Local}(x) \qquad \psi_2 : \mathsf{Local}(x)}{\psi_1 \vee \psi_2 : \mathsf{Local}(x)} \quad \text{(L-Or)}$$

$$\frac{\psi_1 : \mathsf{OC}(x) \qquad \psi_2 : \mathsf{Local}(x)}{\psi_1 \wedge \psi_2 : \mathsf{Local}(x)} \quad \text{(L-And1)}$$

$$\frac{\psi_1 : \mathsf{Local}(x) \qquad \psi_2 : \mathsf{OC}(x)}{\psi_1 \wedge \psi_2 : \mathsf{Local}(x)} \quad \text{(L-And2)}$$

$$\frac{\psi : \mathsf{Local}(x)}{\langle i\rangle\psi : \mathsf{Local}(x)} \quad \text{(L-May)}$$

$$\frac{\psi : \mathsf{Local}(x) \qquad y \neq x}{@_y\,\psi : \mathsf{Local}(x)} \quad \text{(L-At)}$$

$$\frac{\psi : \mathsf{Local}(x) \qquad y \neq x}{\downarrow y\,\psi : \mathsf{Local}(x)} \quad \text{(L-Dow)}$$

Figure 8: A type system for formulas from Fig. 7, identifying polices that are local. The typing rules for inverse modalities $\langle -i\rangle$ and $[-i]$, which closely parallel O-May, O-Mus and L-May, are omitted for brevity.

8. DISCUSSION

We now briefly discuss some issues, alternatives, and ideas not fully developed in this paper. We will not repeat here issues already identified for future work.

Heterogeneous protection state.

The approach to relationship-based access control taken in this paper identifies nodes of a model with principals who may be owners or requesters of resources. So nodes model subjects. But it may be beneficial to let nodes model either *subjects or objects*, so that relationships can also be expressed between subjects and objects.

One idea is to partition the set of nodes S into S_{obj} and S_{subj} so that variables that denote subjects, such as own and req can only be bound to nodes in S_{subj} whereas variables that denote resources can only be bound to nodes in S_{obj}.

An example of the sort of policy one could then write is

$$@_{\mathsf{req}}\,\langle sameFloor\rangle\mathsf{resc} \wedge @_{\mathsf{own}}\,\langle collaborator\rangle\mathsf{req} \wedge$$
$$@_{\mathsf{resc}}\,isPrinter \wedge @_{\mathsf{file}}\,isPDF$$

This policy says that a print job should be granted to req if req is on the same floor as the resource resc, if the resource resc is a printer, if the file file to be printed is in PDF format, and if the owner own is a collaborator of the requester req.

One interesting thing this policy illustrates is the question of whether the ownership is about the file or the printer. Ideally, we want to be able to differentiate such ownership and express it in policy.

One idea is therefore to express ownership also as a binary relation between objects and subjects. In this approach, objects might well have joint ownership. We now could amend the above policy to

$$@_{\mathsf{req}}\,\langle sameFloor\rangle\mathsf{resc}_1 \wedge @_{\mathsf{own}_2}\,\langle collaborator\rangle \wedge$$
$$@_{\mathsf{resc}_1}\,(isPrinter \wedge \langle ownedBy\rangle\mathsf{own}_1) \wedge$$
$$@_{\mathsf{resc}_2}\,(isPDF \wedge @_{\mathsf{resc}_2}\,\langle ownedBy\rangle\mathsf{own}_2)$$

This policy now has two resources, one printer and one file, and each has a possibly different owner. The policy still has the same intent as before, but it also verifies that the file is owned by own_2 whereas the printer is owned by own_1.

Our local model-checking algorithm for HL, and indeed our approach of viewing a policy as a mapping from an object to a formula of HL, are agnostic to, and so directly support, such extensions. Algorithm `MC` works for any formula written in HL. In particular, it works for a fragment

$$\mathsf{HL}(\mathsf{own}_i, \mathsf{resc}_i, \mathsf{req}_i)$$

that generalizes $\mathsf{HL}(\mathsf{own}, \mathsf{req})$ to allow any positive number of owners, resources, and requesters – we already mentioned that the XACML standard allows multiple requesters.

The type of the induced policy function then also changes: each model now has a k-ary relation where k is the number of free variables in the formula that specifies the policy.

Policy composition.

One may think of policies written in HL as just one aspect of control to resources. Different aspects of controlling access could then be combined. For example, in [10] there is a mechanism for determining an appropriate context for evaluating a policy function, where the context is an appropriate model for that policy. Since policies ϕ in HL determine policy functions as well, one can readily use the protection model of [10] with our hybrid logic.

Another example is when we have an attribute-free lan-

guage, e.g. HL with empty AP. Then we may treat each HL policy as a "rule", and then combine rules in a PBel style policy composition language [6], perhaps with rules from other policy languages that refer to attributes.

9. RELATED WORK

Modal logics for access control.

Access control logics for distributed systems can be interpreted as modal logics, with Kripke-style, possible-world semantics. In ABLP [1], every principal is a modality, and states in a model are epistemic states. ICL and its variants [14] can be compiled into formulas in the modal logic S4. Following [10, 12], HL(own, req) formulas are interpreted against models that capture principal attributes and social networks. The nodes of a model denote principals, modalities correspond to relationship types, and accessibility relations express interpersonal relations.

Modal logics for relationship-based access control.

In [12], a modal logic E for relationship-based access control was developed as an extension and improvement of a similar modal logic B. So we focus our discussion on E here.

Its abstract syntax is given by

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle i \rangle \phi \mid \langle -i \rangle \phi \mid \downarrow p\, \phi \mid \phi_1 \otimes \phi_2$$

where p ranges over a set of propositional symbols with a reserved symbol a, and i ranges over a set of labels. In [12], the operator $\downarrow p$ is actually written $@_p$ but we chose the former notation here, as it will aid our comparative discussion.

The models for E are similar to those for HL. Semantically, propositions are treated in E like nominals in hybrid logic: they are true in exactly one node. Symbol a represents the requester of an access. The semantics of the $\downarrow p$ operator is actually that of the $\downarrow n$ operator in hybrid logic, except that $\downarrow \mathsf{a}\, \phi$ is interpreted as $\bot$. Intuitively, $\downarrow p\, \phi$ identifies p with the owner of the object.

Lastly, operator $\phi \otimes \psi$ holds if one can separate model M into two parts of node sets that only overlap for the owner and requester, and where ϕ holds in one of these parts and ψ holds in the other one. This operator crucially increased the expressiveness of **B** so that, e.g., thresholds on the number of specific successors of a node can be expressed. Since one can encode graded modalities and similar counting mechanisms in HL, we do not need to add such an operator to our logic.

We view this fact as a big advantage of using HL instead of E. For $\phi \otimes \psi$ incurs a seemingly unavoidable exponential penalty, since there are exponentially many partitions of the set of nodes that need to be considered in its evaluation. The evaluation of HL over models, however, is linear in the size of the model and formula with appropriate caching in place.

Another advantage of HL is the greater flexibility of atomic formulas: HL offers nominals, variables, and propositions (i.e. attributes). Further, the operator $@_t$ is very useful as it specifies where a policy should be evaluated. In fact, by promoting HL(own, req) we suggest most policies would be Boolean combinations of policies of form $@_t\, \psi$.

Hybrid logics for access control.

We are not aware of much other work on using hybrid logic for access control. But in [7], a logical framework is used to design an authorization logic for time-dependent access-control decisions. The logic contains a variant of the $@_t$ operator, $A@I$, which allows the relativization of the truth of proposition A to the time interval I.

Local-ness in ReBAC policies.

Local-ness ensures that change of authorization decision is due to a change in "connectivity" between the owner and the requester of an object. The notion was originally formulated in [2] (an extension of [11]), in which a local policy is one such that, if there is a change of authorization decision due to the introduction of one new edge, then the new edge must be connected to both the owner and the requester. The definition was based on social networks that are undirected graphs, with no edge labels.

In [8], the definition was generalized to account for the introduction of "one or more" edges. The two definitions can be shown to be equivalent. In [12], the definition was adapted to account for social networks that are directed and poly-relational, such that the definition in [8] is merely a special case. In this work we adopt a formulation of local-ness that makes explicit the kind of graph structures that local policies cannot distinguish. The new formulation is equivalent to that of [12] (Proposition 3). The definition of i-local policies in the Appendix A is again a "backward compatible" generalization that accounts for policies of arbitrary arity.

The notion of properly local policies was originally formulaed in [9] (an extension of [8]).

10. CONCLUSIONS

In this paper we have proposed the use of hybrid logic for the specification and enforcement of access-control decisions in the relationship-based approach to access control.

Concretely, we presented a fragment of hybrid logic that is customized to the needs of relationship-based access control. We demonstrated that the models of that hybrid logic are appropriate as models of protection states in relationship-based access control. We showed how the semantics of hybrid logic on such models gives meaning to access-control policies written in that logic.

Then we discussed how this semantics can be implemented as a policy-decision point, via a local model-checking algorithm. Next, we featured numerous examples of policies and showed how they can be elegantly specified in our hybrid logic. Importantly, we showed how it can express graded modalities such as "at least three friends".

We then transferred results from hybrid logic to our setting, showing that all policies written in our hybrid logic are invariant under so called generated sub-models, and that binder-free policies from our hybrid logic are invariant under hybrid bisimulation.

To understand better connections with related work, we identified an attribute-free fragment of our hybrid logic, via a static type system, in which only policies can be specified that are *relational* in a technical sense from the literature.

We also stressed that the scope of our approach is not confined to having sole owners, resources, and requesters. We finally concluded the paper with a discussion of related work and of how our approach improves on it.

11. REFERENCES

[1] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems.

ACM Transactions on Programming Languages and Systems, 15(4):706–734, Sept. 1993.

[2] M. Anwar, Z. Zhao, and P. 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, July 2010. Submitted for review.

[3] C. Areces. *Logic Engineering. The Case of Description and Hybrid Logics*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands, October 2000.

[4] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logics*. Elsevier, 2006.

[5] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*. Elsevier, 2007.

[6] G. Bruns and M. Huth. Access control via Belnap logic: Intuitive, expressive, and analyzable policy composition. *ACM Trans. Inf. Syst. Secur.*, 14(1):9, 2011.

[7] H. DeYoung. A logic for reasoning about time-dependent access control policies. Senior Research Thesis CMU-CS-08-131, School of Computer Science, Carnegie Mellon University, 20 May 2008.

[8] P. W. L. Fong. Preventing Sybil attacks by privilege attenuation: A design principle for social network systems. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy (S&P'11)*, pages 263–278, Oakland, CA, USA, May 2011.

[9] P. W. L. Fong. Preventing Sybil attacks by privilege attenuation: A design principle for social network systems. Technical Report 2011-995-07, Department of Computer Science, University of Calgary, Mar. 2011.

[10] P. W. L. Fong. Relationship-based access control: protection model and policy language. In *CODASPY*, pages 191–202, 2011.

[11] P. W. L. Fong, M. Anwar, and Z. 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 *LNCS*, pages 303–320, Saint Malo, France, Sept. 2009.

[12] P. W. L. Fong and I. Siahaan. Relationship-based access control policies and their policy languages. In *SACMAT*, pages 51–60, 2011.

[13] M. Franceschet and M. de Rijke. Model checking hybrid logics (with an application to semistructured data). *J. Applied Logic*, 4(3):279–304, 2006.

[14] D. Garg and M. Abadi. A modal deconstruction of access control logic. In *FOSSACS'2008*, volume 4962 of *LNCS*, pages 216–230, 2008.

[15] C. E. Gates. Access control requirements for web 2.0 security and privacy. In *Proc. of IEEE Web 2.0 Privacy and Security Workshop (W2SP'07)*, Oakland, California, May 2007.

[16] T. Moses. extensible access control markup language (XACML) version 2.0. Technical report, OASIS Standard, 1 February 2005. Specification document.

APPENDIX

A. PROOF OF TYPE SOUNDNESS

This appendix sketches the proof of Theorem 2.

The introduction of variables (x) causes an authorization decision to depend on the relationship not only between the owner and the requester, but also among other individuals represented by variables. To accommodate this, we generalize the notion of a semantic policy as follows.

DEFINITION 14. *For $k \geq 2$, a k-ary policy function P is a family $(P(M))_M$ of k-ary relations, indexed by models M, where $P(M) \subseteq (S_M)^k$.*

Note that our ultimate goal is to express binary policies. Given $(s_0, s_1, \ldots, s_k)$ in $(S_M)^{k+1}$, we interpret s_0 to be one end of the binary relation (e.g., the owner). We call this end the *source*. The other k components correspond to individuals named by variables (x). One of the named individuals will be selected to represent the other end of the binary relation (e.g., the requester). We call this other end the *target*. The other $k-1$ individuals are called *bridges*.

We write $\bar{x}$ to denote a sequence of distinct variables $x_1, \ldots, x_k$. We write $\psi(\bar{x})$ to denote the coupling of a formula ψ with the sequence $\bar{x}$, such that every free variable of ψ appears in $\bar{x}$. Without loss of generality, we assume that every binder $\downarrow x$ binds a distinct variable in ψ.

DEFINITION 15. *Given a formula ψ as defined in Figure 7, a variable sequence $\bar{x} = x_1, \ldots, x_k$ such that* $\mathsf{freevars}(\psi) \subseteq \{\bar{x}\}$*, the policy function* $\mathsf{policy}[\psi(\bar{x})]$ *is defined as follows:*

$$\mathsf{policy}[\psi(\bar{x})](M) \stackrel{\text{def}}{=} \{(s_0, s_1, \ldots, s_k) \in (S_M)^{k+1} \mid M, s_0, [x_1 = s_1, \ldots, x_k = s_k] \models \psi\}$$

We present here an alternative but equivalent formulation of local policies, in preparation for further generalization.

DEFINITION 16. *Let M and N be models.*

1. *We write $M \subseteq N$ if $S_M \subseteq S_N$ and, for all $i \in I$, $R_i^M \subseteq R_i^N$.*
2. *We write $N - M$ for the binary relation on S_N defined by $\{(s,t) \in S_N \times S_N \mid \exists i \in I\colon (s,t) \in R_i^N \setminus R_i^M\}$.*

PROPOSITION 3. *A binary policy function P is local iff the following holds: for all models M and N such that $M \subseteq N$, and all $(s,t) \in S_M \times S_M$, if $P(M)(s,t) \neq P(N)(s,t)$ then there is $(s',t') \in N - M$ such that $s \overset{N}{\rightsquigarrow} s'$ and $t' \overset{N}{\rightsquigarrow} t$.*

The condition stated in this proposition is in fact the definition of local policies in [12]. The following generalization of local policies is based on this formulation.

DEFINITION 17. *A $(k+1)$-ary policy P is* i-local *$(1 \leq i \leq k)$ iff the following holds:*

> *For all models M, N with $M \subseteq N$, and all $(s_0, s_1, \ldots, s_k) \in (S_M)^{k+1}$, if $s_0 \overset{M}{\rightsquigarrow} s_j$ for every $j \in \{1, \ldots, k\} \setminus \{i\}$, then $P(M)(s_0, s_1, \ldots, s_k) \neq P(N)(s_0, s_1, \ldots, s_k)$ implies that there is a pair $(s', t') \in N - M$ such that $s_0 \overset{N}{\rightsquigarrow} s'$ and $t' \overset{N}{\rightsquigarrow} s_i$.*

Note that a binary policy P is local iff P is 1-local.

This definition of i-local policy generalizes the previous definition of local policy in two ways. First, the index i

essentially identifies target of the binary relation (recall s_0 is the source). The rest of the components s_j (where $1 \leq j \leq k$ and $j \neq i$) are bridges. Second, the definition defines a notion of "conditional" local-ness. Under the condition that the bridges are connected to the source, the requirements of local-ness apply.

To address the issue that local-ness is not preserved by relational composition, we need to strengthen the notion of local-ness. The additional requirement is captured in the following definition, which generalizes a similar definition given in [9] – which focuses on monotonic policies.

DEFINITION 18. *1. A $(k+1)$-ary policy P is* i-proper *$(1 \leq i \leq k)$ iff the following holds: For every model M, and every $(s_0, s_1, \ldots, s_k) \in (S_M)^{k+1}$, if $s_0 \overset{M}{\rightsquigarrow} s_j$ for every $j \in \{1, \ldots, k\} \setminus \{i\}$, then $P(M)(s_0, s_1, \ldots, s_k)$ implies $s_0 \overset{M}{\rightsquigarrow} s_i$.*

2. A policy P is i-properly local *iff P is both i-local and i-proper.*

Properly local policies form a family that is closed under both relational composition and positive boolean combinations. Also, no properly local policy is semantically equivalent to $\mathsf{p}[\top]$.

One important observation is that the conjunction of a properly local policy with a so-called "*owner-checkable policy*" [12] results in a policy that is also properly local. This provides a rich mechanism for composing properly local policies out of components that need not be properly local.

DEFINITION 19. *A $(k+1)$-ary policy P is* i-owner-checkable (i-OC) *$(1 \leq i \leq k)$ iff the following holds:*

For all models M, N such that $M \subseteq N$, and all $(s_0, s_1, \ldots, s_k) \in (S_M)^{k+1}$, if $s_0 \overset{M}{\rightsquigarrow} s_j$ for every $j \in \{1, \ldots, k\} \setminus \{i\}$, then $P(M)(s_0, s_1, \ldots, s_k) \neq P(N)(s_0, s_1, \ldots, s_k)$ implies that there is a pair $(s', t') \in N - M$ such that $s_0 \overset{N}{\rightsquigarrow} s'$.

The definition of an OC policy is a weakening of the definition of local policy. When there is a change in authorization decision between the owner and the requestor, a local policy ensures that the change must be due to the introduction of an edge that connects both the owner and the requestor. An OC policy requires less: it only ensures that the edge is connected to the owner[1].

PROPOSITION 4. *Suppose $\bar{x} = x_1, \ldots, x_k$, and $1 \leq i \leq k$.*

1. *If $\mathsf{policy}[\psi(\bar{x})]$ is i-OC, and $\{\bar{x}\} \subseteq \{\bar{y}\}$ such that $x_i = y_j$, then $\mathsf{policy}[\psi(\bar{y})]$ is j-OC.*
2. *Every i-properly local policy is also i-OC.*
3. *The policies $\mathsf{policy}[\top(\bar{x})]$, $\mathsf{policy}[\bot(\bar{x})]$ and $\mathsf{policy}[y(\bar{x})]$ are i-OC.*
4. *If both $\mathsf{policy}[\psi_1(\bar{x})]$ and $\mathsf{policy}[\psi_2(\bar{x})]$ are i-OC, then $\mathsf{policy}[(\neg\psi_1)(\bar{x})]$, $\mathsf{policy}[(\psi_1 \vee \psi_2)(\bar{x})]$ and $\mathsf{policy}[(\psi_1 \wedge \psi_2)(\bar{x})]$ are i-OC.*
5. *If $\mathsf{policy}[\psi(\bar{x})]$ is i-OC, then $\mathsf{policy}[(\langle i \rangle \psi)(\bar{x})]$, $\mathsf{policy}[(\langle -i \rangle \psi)(\bar{x})]$, $\mathsf{policy}[([i]\psi)(\bar{x})]$ and $\mathsf{policy}[([-i]\psi)(\bar{x})]$ are i-OC.*
6. *If $\mathsf{policy}[\psi(\bar{x})]$ is i-OC, and $j \neq i$, then $\mathsf{policy}[(@_{x_j}\, \psi)(\bar{x})]$ is i-OC.*
7. *If $\mathsf{policy}[\psi(\bar{x}, y)]$ is i-OC, then $\mathsf{policy}[(\downarrow y\, \psi)(\bar{x})]$ is i-OC.*

The above proposition can be succinctly summarized in two points. First, properly local policies are OC. Second, the syntactic construction of ψ in Figure 7 mostly preserves owner-checkability, so long as the variable y in $@_y$ and $\downarrow y$ is not the target variable.

PROPOSITION 5. *Suppose $\bar{x} = x_1, \ldots, x_k$, and $1 \leq i \leq k$.*

1. *If $\mathsf{policy}[\psi(\bar{x})]$ is i-properly local, and $\{\bar{x}\} \subseteq \{\bar{y}\}$ such that $x_i = y_j$, then $\mathsf{policy}[\psi(\bar{y})]$ is j-properly local.*
2. *The policies $\mathsf{policy}[\bot(\bar{x})]$ and $\mathsf{policy}[x_i(\bar{x})]$ are i-properly local.*
3. *If both $\mathsf{policy}[\psi_1(\bar{x})]$ and $\mathsf{policy}[\psi_2(\bar{x})]$ are i-properly local, then $\mathsf{policy}[(\psi_1 \vee \psi_2)(\bar{x})]$ is i-properly local.*
4. *If $\mathsf{policy}[\psi_1(\bar{x})]$ is i-properly local, and $\mathsf{policy}[\psi_2(\bar{x})]$ is i-OC, then both $\mathsf{policy}[(\psi_1 \wedge \psi_2)(\bar{x})]$ and $\mathsf{policy}[(\psi_2 \wedge \psi_1)(\bar{x})]$ are i-properly local.*
5. *If $\mathsf{policy}[\psi(\bar{x})]$ is i-properly local, then $\mathsf{policy}[(\langle i \rangle \psi)(\bar{x})]$ and $\mathsf{policy}[(\langle -i \rangle \psi)(\bar{x})]$ are i-properly local.*
6. *If $\mathsf{policy}[\psi(\bar{x})]$ is i-properly local, and $j \neq i$, then $\mathsf{policy}[(@_{x_j}\, \psi)(\bar{x})]$ is i-properly local.*
7. *If $\mathsf{policy}[\psi(\bar{x}, y)]$ is i-properly local, then $\mathsf{policy}[(\downarrow y\, \psi)(\bar{x})]$ is i-properly local.*

The only base cases in the above proposition are $\bot$ and the positive testing of the target variable x. One of the most important highlights in the above proposition is that a conjunction is properly local so long as one of its conjuncts is properly local, and the other conjunct can simply be OC. Yet, for a disjunction to be properly local, both disjuncts must be properly local. A second important highlight is that $\langle i \rangle$ and $\langle -i \rangle$ preserves proper local-ness. This addresses the crux of the problem of relational composition. Lastly, the variable y in $@_y$ and $\downarrow y$ must not be the target variable if proper local-ness is to be preserved.

Theorem 2 is a corollary of the following result, which in turn follows directly from Propositions 4 and 5.

THEOREM 3. *Suppose $\bar{x} = x_1, \ldots, x_k$, and $1 \leq i \leq k$.*

1. *If $\psi : \mathsf{OC}(x_i)$, then $\mathsf{policy}[\psi(\bar{x})]$ is i-OC.*
2. *If $\psi : \mathsf{Local}(x_i)$, then $\mathsf{policy}[\psi(\bar{x})]$ is i-properly local.*

[1]Our definition of 1-OC policy is a minor generalization of the definition of OC policy in [12]. The latter is restricted to topology-based policies. Our definition relaxes this restriction. The definition of [12] (without the topology-based requirement) is the following: A binary policy function P is OC iff, for all models M and N, and all $s, t \in S_M \cap S_N$, if $\mathsf{C}(M, s, t) = \mathsf{C}(N, s, t)$, then $P(M)(s, t) = P(N)(s, t)$. Here, $\mathsf{C}(M, s, t)$ is the relational structure $(S, \{R_i \subseteq S \times S \mid i \in I\})$ with the components below:

$$S \overset{\text{def}}{=} \{s, t\} \cup \{s' \in S_M \mid s \overset{M}{\rightsquigarrow} s'\}$$

$$R_i \overset{\text{def}}{=} R_i^M \cap (S \times S)$$

Using an argument similar to the proof of Proposition 3, one can show that the definition above is equivalent to 1-OC.