Implementing a Calculus for Distributed Access Control in Higher Order Logic and HOL *

Thumrongsak Kosiyatrakul, Susan Older, Polar Humenn, and Shiu-Kai Chin

Department of Electrical Engineering and Computer Science & Systems Assurance Institute, Syracuse University, Syracuse, New York 13244, USA

Abstract. Access control—determining which requests for services should be honored or not—is particularly difficult in networked systems. Assuring that access-control decisions are made correctly involves determining identities, privileges, and delegations. The basis for making such decisions often relies upon cryptographically signed statements that are evaluated within the context of an access-control policy.

An important class of access-control decisions involves *brokered services*, in which intermediaries (brokers) act on and make requests on behalf of their clients. Stock brokers are human examples; electronic examples include the web servers used by banks to provide the online interface between bank clients and client banking accounts. The CORBA (Common Object Request Broker Architecture) CSIv2 (Common Secure Interoperability version 2) protocol is an internationally accepted standard for secure brokered services [2]. Its purpose is to ensure service requests, credentials, and access-control policies have common and consistent interpretations that lead to consistent and appropriate access-control decisions across potentially differing operating systems and hardware platforms. Showing that protocols such as CSIv2 fulfill their purpose requires reasoning about identities, statements, delegations, authorizations, and policies and their interactions.

To meet this challenge, we wanted to use formal logic to guide our thinking and a theorem prover to verify our results. We use a logic for authentication and access control [5, 3, 8] that supports reasoning about the principals in a system, the statements they make, their delegations, and their privileges. To *assure* our reasoning is correct, we have implemented this logic as a definitional extension to the HOL theorem prover [4]. We describe this logic, its implementation in HOL, and the application of this logic to brokered requests in the context of the CORBA CSIv2 standard.

1 Introduction

The rapid growth of wired and wireless networks has resulted in distributed computing on a global scale where services are available anywhere at anytime. While

* Partially supported by the New York State Center for Advanced Technology in Computer Applications and Software Engineering (CASE)

global access to information and services offers great convenience, it also carries security risks that include inappropriate disclosure of information, potential loss and corruption of data, identity theft, and fraud. These security risks are rooted in the inadequate verification of identity, authorization, and integrity.

To understand both the importance and the challenges of trustworthy systems in a global scale, consider the following scenario:

> John Lee, an American tourist, is traveling in Asia on vacation twelve time zones away from home. John has an accident and goes to a local hospital for treatment. Dr. Gupta, the attending physician, concludes that immediate intervention is required, but to provide safe and effective treatment, she needs access to Mr. Lee's medical records located halfway around the world. Dr. Gupta contacts her hospital's medical records administrator, Mr. Kumar, to obtain access to Mr. Lee's records. Mr. Kumar electronically retrieves Mr. Lee's medical records (and *only* his medical records) from his doctor's computers within minutes, despite the office being closed.

In the above scenario, there are three primary concerns: (1) John Lee should be assured that his private medical information is accessible only to the proper authorities under specific circumstances and that the information is correct and available when needed; (2) the hospital should be assured of immediate and timely access in an emergency and that the information is correct; and (3) Mr. Lee's doctor's office should be free from liability concerns about the improper release of records or incorrect data. These goals can be achieved by the correctness of access-control decisions and delegations. To assure the correctness of access-control decisions, several components are required: a protocol for handling brokered requests; consistent and predictable interpretations of requests; and a logically sound theory for reasoning about delegations and access-control decisions.

The specific protocol we consider in this paper is the Common Object Request Broker Architecture (CORBA) Common Secure Interoperability Version 2 Protocol (CSIv2). CORBA is an open standard that supports component-based software designs and services at the middleware level [1]. CSIv2 is an accepted CORBA standard in support of authentication of client identity [2]. The CSIv2 protocol was designed specifically to augment existing authentication technology (e.g., SSL) with authorization and delegation information that can be used to distinguish brokers from the clients who originate requests.

We focus on reasoning about access-control decisions and delegations related to this protocol. We introduce a specialized modal logic that originally appeared in a series of papers by Abadi and colleagues [5, 3, 8]. This logic supports reasoning about the statements made by principals (i.e., the actors in a system), their beliefs, their delegations, and their authorizations. To ensure the soundness of the logic, we embedded this logic into the HOL system as a definitional extension to verify the correctness of the formal definitions. A benefit provided by this embedding is confidence in the correctness and soundness of the axioms of the logic.

The rest of this paper is organized as follows. Section 2 describes the CORBA CSIv2 protocol. Section 3 describes the syntax and semantics of Abadi and colleagues' calculus [5, 3, 8] that is used to reason about principals and their belief. In Section 4, we illustrate how that logic can be used to describe aspects of brokered requests. In Section 5, we describe how the calculus is implemented as a definitional extension to the higher-order logic of the HOL theorem prover. Section 6 shows how to use our theory to prove a property about the access-control example from Section 4. Finally, our conclusions are presented in Section 7.

2 Protocol for Brokered Requests

The CSIv2 protocol is designed for passing on requests from clients (service requestors) to targets (service providers). We describe the protocol in terms of *principals*, who are the subjects of the system: they make requests, grant privileges, and delegate or endorse other principals to act on their behalf. When a service provider receives a request, she must determine who is actually making the request: the principal who transmitted it to her may be making it on his own behalf or on behalf of another principal. Furthermore, the provider must also determine whether that principal is authorized to make that particular request.

In the CSIv2 protocol, there are potentially four principals who participate in any given request: (1) the target service provider (*Carol*), (2) a transport principal (*Default* or *Tony*) who delivers the request to Carol, (3) a client to be authenticated (*Clyde*), and (4) a principal whose name (*Alice* or *Anon*) appears in an identity token. Note that we are using different names to highlight the different aspects of a request—in practice, it is possible for these different components to refer to the same entity.

Every request that Carol receives is transmitted over either TCP/IP or SSL. When Carol receives the request over TCP/IP, the transport principal is *Default*: Carol does not authenticate this principal, but merely assumes the request was sent by the default entity that always transmits requests to her. This scenario is appropriate for situations in which the network is secure (e.g., private networks) and for which Carol assumes that the request arrived through a pathway she trusts. When, instead, Carol receives the request over SSL, she authenticates the transport principal (who we identify as *Tony* for expository purposes) using a public-key certificate.

To understand the purpose of the named principals Clyde, Alice, and Anon, it is helpful to first understand the structure of CSIv2 requests. These requests may include a Security Attribute Service (SAS) data structure, which holds userid/password pairs and identity tokens. The SAS data structure can be viewed as a three-tuple $\langle CA, IA, SA \rangle$, with the following components:

- CA (if present) contains *client authentication* information, such as userid and passwords.
- IA is an *identity assertion*, which (if present) identifies the client of the request.

– SA contains any applicable *security attributes*, which indicate various privileges that the relevant principals may have.

The CSIv2 authors have not yet standardized the interpretation of the information contained in the SA component, and thus we focus our attention on the CA and IA components for now. In total, there are six possible pairs $\langle CA, IA \rangle$ to consider. The CA component may either be empty (in which case no principal is associated with CA) or contain a userid/password pair $\langle \mathit{Clyde}, P \rangle$ (in which case the principal Clyde is associated with CA). There are three possibilities for the IA component: it may be empty, or it may contain an identity token for a particular principal Alice or for the Anonymous principal.

When the target provider Carol receives a request r with an associated SAS data structure, she always interprets it relative to the same ordering on principals: the transport principal (either Default or Tony) first, followed by (if present) the principal associated with CA (Clyde), followed by (if present) the principal associated with IA (Alice or Anon). Thus, if an SSL request r identifies Clyde as the CA principal and Alice as the IA principal, then Carol interprets this request as

$$\mathit{Tony}\ \mathsf{says}\ (\mathit{Clyde}\ \mathsf{says}\ (\mathit{Alice}\ \mathsf{says}\ r)),$$

and she considers Alice to be the *invocation principal* (i.e., client). In contrast, a TCP/IP request r that identifies Clyde as the CA principal and contains no IA component is interpreted as

$$\mathit{Default}\ \mathsf{says}\ (\mathit{Clyde}\ \mathsf{says}\ r),$$

and Clyde is the invocation principal. Table 1 enumerates the twelve cases for requests that are made in association with the SAS data structure.

Connection Method	Transport Principal	CA	IA	Invocation Principal	Carol's Interpretation
TCP/IP	Default	None	None	Default	Default says r
TCP/IP	Default	(Clyde,PW)	None	Clyde	Default says Clyde says r
SSL	Tony	None	None	Tony	Tony says r
SSL	Tony	(Clyde,PW)	None	Clyde	Tony says Clyde says r
TCP/IP	Default	None	Alice	Alice	Default says Alice says r
TCP/IP	Default	(Clyde, PW)	Alice	Alice	Default says Clyde says Alice says r
SSL	Tony	None	Alice	Alice	Tony says Alice says r
SSL	Tony	(Clyde, PW)	Alice	Alice	Tony says Clyde says Alice says r
TCP/IP	Default	None	Anon	Anon	Default says Anon says r
TCP/IP	Default	(Clyde, PW)	Anon	Anon	Default says Clyde says Anon says r
SSL	Tony	None	Anon	Anon	Tony says Anon says r
SSL	Tony	(Clyde, PW)	Anon	Anon	Tony says Clyde says Anon says r

Table 1. Interpretation of CSIv2 requests

The fourth case in the table—and the associated statement $\mathit{Tony}\ \mathsf{says}\ \mathit{Clyde}\ \mathsf{says}\ r$—can be used to describe a situation in which a bank customer (here, Clyde) connects with a web server to perform an online-banking operation. The web server passes on Clyde's username, password, and request r to an online

banking server (Carol) over SSL; because the request is transmitted over SSL, Carol authenticates the web server as Tony, rather than identifying the web server as Default. The CSIv2 protocol specifies how the information is to be interpreted by the target principal.

Because delegated authority must be carefully controlled and limited to prevent fraud, reasoning about delegation must be done precisely, accurately, and consistently. To meet these needs we use a specialized calculus and modal logic for reasoning about brokered requests and delegations. In the next section, we give an overview of this logic; we then revisit the banking scenario in Section 4.

3 Calculus for Reasoning About Brokered Requests

In a series of papers [5, 3, 8], Abadi and colleagues introduced a logic for authentication and access control for distributed systems. This logic incorporates a calculus of principals into a standard multi-agent modal logic: the calculus of principals provides a mechanism to describe ordering relationships among principals, while the modal logic provides a mechanism for reasoning about principals' credentials and the statements they make. The result is a logic that supports reasoning about access control, delegation, and trust.

3.1 Calculus of Principals

Syntactically, the calculus of principals is very simple. We assume the existence of a countable set containing *simple principal names* and ranged over by the meta-variable A. The set of *principal expressions* (ranged over by P and Q) has an abstract syntax given by the following grammar:

$$P ::= A \mid P \wedge Q \mid P|Q \mid P \text{ for } Q$$

Intuitively, $P \wedge Q$ denotes a principal whose statements correspond precisely to those statements that P and Q jointly make. $P|Q$ denotes a principal whose statements correspond to those that P attributes to Q. Finally, P for Q denotes a principal whose statements correspond to those that P is authorized to attribute to Q. We consider $P \wedge Q$ to be a stronger (i.e., more trustworthy) principal than either P or Q because of the consensus. Likewise, we consider P for Q to be a stronger principal than $P|Q$ because of the authorization.

Semantically, we interpret principal expressions over a multiplicative semilattice semigroup (MSS). An MSS $(S, \curlywedge, \|)$ is a set S equipped with two binary operations ($\curlywedge$and $\|$): $\curlywedge$ is idempotent, commutative and associative, while $\|$ is associative and distributes over $\curlywedge$ in both arguments. For S-elements x and y, the element $x \curlywedge y$ is the *meet* of x and y. Thus the operator $\curlywedge$ induces an ordering $\leq$ on S as follows: $x \leq y$ if and only if $x = x \curlywedge y$.

As a common (and useful) example of a MSS, consider any set T, and let T_R be a set of binary relations over T that is closed under union ($\cup$) and relational composition ($\circ$). It is straightforward to verify that $(T_R, \cup, \circ)$ satisfies the MSS

axioms. In this case, the ordering $\leq$ amounts to the superset relation: $r_1 \leq r_2$ if and only if $r_1 = r_1 \cup r_2$, which is true precisely when $r_1 \supseteq r_2$.

Every MSS can be used to provide an interpretation for principal expressions. Specifically, consider any structure $\mathcal{M} = \langle \mathcal{P}, J \rangle$, where $\mathcal{P}$ is an MSS and J is a total function that maps each simple principal name to an element of the MSS $\mathcal{P}$. We can extend J to a meaning function $\tilde{J}$ that provides an interpretation for *all* principal expressions, as follows:

$$\tilde{J}(A) = J(A), \qquad \tilde{J}(P \wedge Q) = \tilde{J}(P) \curlywedge \tilde{J}(Q), \qquad \tilde{J}(P|Q) = \tilde{J}(P) \| \tilde{J}(Q)$$

Abadi and colleagues suggest that the principal P for Q can be viewed as syntactic sugar for $(P|Q) \wedge (D|Q)$, where D is a (fictional) distinguished principal that validates P's authorization to make statements on Q's behalf.

3.2 Modal Logic

The calculus of principals provides a mechanism to describe ordering relationships among principals. To reason about principals' credentials and access control, we turn to modal logic.

Syntax We assume the existence of a countable set of *propositional variables*, ranged over by the meta-variables p and q. This set, along with formulas of form "P speaks_for Q" and "P says φ" that relate principals and statements, forms the basis for a set **LogExp** of *logical formulas*. The abstract syntax for logical formulas, ranged over by φ is given by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \,\&\, \varphi_2 \mid \varphi_1 \text{ or } \varphi_2 \mid \varphi_1 \supset \varphi_2 \mid \varphi_1 \iff \varphi_2 \mid P \text{ says } \varphi \mid P \text{ speaks\_for } Q$$

We will see that, semantically, the formula P speaks_for Q holds when $P \leq Q$ is true in the underlying calculus of principals. It follows that speaks_for is monotonic with respect to both $\wedge$ and $|$.

Logical rules Having identified the syntax of logical expressions, we now introduce a collection of logical rules for manipulating them. In particular, we define a derivability predicate $\vdash$ over logical formulas: we write $\vdash \varphi$ provided that the formula φ is derivable from the collection of rules. This collection of rules can be split into two groups: those standard for all modal logics and those relating the modal logic to the calculus of principals. These rules appear in Figure 1.

Semantics The logical rules are useful for deducing relationships among principals and for reasoning about issues of delegation and trust. However, as given, they merely provide rules for manipulating syntactic expressions: there is no *a priori* reason to trust in their consistency. We therefore must introduce a semantic framework in which to prove their *soundness*.

We have already seen that MSSs can provide suitable interpretations for the calculus of principals. For simplicity of exposition, in the following discussion we

$$\frac{}{\vdash \varphi} \quad \text{if } \varphi \text{ is an instance of a propositional-logic tautology}$$

$$\frac{\vdash \varphi \quad \vdash \varphi \supset \varphi'}{\vdash \varphi'} \qquad \frac{\vdash \varphi}{\vdash P \text{ says } \varphi} \quad \text{(for all } P\text{)}$$

$$\frac{}{\vdash (P \text{ says } (\varphi \supset \varphi')) \supset (P \text{ says } \varphi \supset P \text{ says } \varphi')}$$

$$\frac{}{\vdash \varphi} \quad \text{if } \varphi \text{ a valid formula of the calculus of principals}$$

$$\frac{}{\vdash (P \wedge Q) \text{ says } \varphi \iff (P \text{ says } \varphi) \;\&\; (Q \text{ says } \varphi)}$$

$$\frac{}{\vdash (P|Q) \text{ says } \varphi \iff P \text{ says } Q \text{ says } \varphi}$$

$$\frac{}{\vdash (P \text{ speaks\_for } Q) \supset ((P \text{ says } \varphi) \supset (Q \text{ says } \varphi))} \text{ (for all } \varphi\text{)}$$

Fig. 1. Logical rules for the derivability predicate $\vdash$

restrict ourselves to algebras of binary relations; however, arbitrary MSSs can be used as the underlying model for principals with some additional overhead (for example, see [3]).

We use *Kripke structures* to provide interpretations for logical formulas. A Kripke structure $\mathcal{M} = \langle W, w_0, I, J \rangle$ comprises a set W of *possible worlds*; a distinguished element $w_0 \in W$; an interpretation function I, which maps each propositional variable to a subset of W; and an interpretation function J, which maps each principal name to a binary relation over W (i.e., a subset of $W \times W$). Intuitively, $I(p)$ is the set of worlds in which the propositional variable p is true, and $J(P)$ is the accessibility relation for principal P: if (w, w') is in $J(P)$, then principal P cannot distinguish w' from w. One must be careful with this intuition, however: there is no guarantee that the relation $J(P)$ is symmetric (or even reflexive or transitive) for an arbitrary principal P, as there is no *a priori* expectation that an arbitrary principal will be rational.

As we saw previously, we can extend J to a meaning function $\tilde{J}$ that operates over arbitrary principal expressions:

$$\tilde{J}(A) = J(A), \qquad \tilde{J}(P \wedge Q) = \tilde{J}(P) \cup \tilde{J}(Q)$$
$$\tilde{J}(P|Q) = \tilde{J}(Q) \circ \tilde{J}(P) = \{(w, w'') \mid \exists w'.(w, w') \in \tilde{J}(P) \;\&\; (w', w'') \in \tilde{J}(Q)\}$$

Finally, we can define a meaning function $\mathcal{E}_{\mathcal{M}} : \mathbf{LogExp} \rightarrow 2^W$ that gives the set of worlds in which a formula is true:

$$\begin{aligned}
\mathcal{E}_{\mathcal{M}}[\![p]\!] &= I(p) \\
\mathcal{E}_{\mathcal{M}}[\![\neg\varphi]\!] &= W - \mathcal{E}_{\mathcal{M}}[\![\varphi]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \ \&\ \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
&\vdots \\
\mathcal{E}_{\mathcal{M}}[\![P \text{ says } \varphi]\!] &= \{w \mid \tilde{J}(P)w \subseteq \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]\} \\
&= \{w \mid \ \{w' \mid (w,w') \in \tilde{J}(P)\} \subseteq \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]\} \\
\mathcal{E}_{\mathcal{M}}[\![P \text{ speaks_for } Q]\!] &= \begin{cases} W, \text{ if } \tilde{J}(Q) \subseteq \tilde{J}(P) \\ \emptyset, \text{ otherwise} \end{cases}
\end{aligned}$$

We write $\mathcal{M}, w \models \varphi$ provided that $w \in \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$. We write $\mathcal{M} \models \varphi$ provided that $\mathcal{M}, w_0 \models \varphi$; this relationship is pronounced "$\mathcal{M}$ satisfies φ". Finally, we say that φ is *valid* (and write $\models \varphi$) if all Kripke structures $\mathcal{M}$ satisfy φ.

The logical rules of Figure 1 are *sound* with respect to the Kripke semantics: for every formula φ, $\vdash \varphi$ implies that $\models \varphi$. Therefore, every formula that can be derived from the logical rules is satisfied in all Kripke structures.

4 CORBA Example in the Calculus

We now return to the online-banking scenario introduced in Section 2, changing the names of the actors for expository purposes:

> A banking customer Bob uses a web browser to access his online banking account to pay a bill. The web browser (WB) requires Bob to give his username and password ($\langle Bob, pwd \rangle$), which the browser passes along with Bob's bill-payment request (r) to the online-banking server (S) via a CORBA CSIv2 request.

In terms of Table 1, WB is playing the part of Tony (i.e., a transport principal who uses SSL), and Bob is cast as Clyde (i.e., the client-authentication field contains Bob's userid and password). The online-banking server is the target service provider, and thus serves as Carol. If we assume that there is no identity token included with the request, then this scenario again conforms to the fourth case of the table.

The banking server interprets the request as two pieces of information:

$$WB \text{ says } Bob \text{ says } r \tag{1}$$

$$WB \text{ says } (\langle Bob, pwd \rangle \text{ speaks\_for } Bob) \tag{2}$$

Thus, from the banking server's perspective, WB is making two claims: (1) that Bob wants to perform action r, and (2) that the username-password pair belongs to Bob.

Now suppose that banking server has an access-control policy that allows WB for$_S$ Bob to perform the action r: that is, r can be performed only if S determines that WB is working on Bob's behalf. In the Abadi calculus, WB for Bob is syntactic sugar for a principal $WB|Bob \wedge D|Bob$, where D is a (fictional) delegation authority. In an open system, however, there may be several such (possibly nonfictional) authorities. Thus, we generalize this approach by subscripting the for operator with the name of a *specific* authority, in this case the banking server S itself.

The online-banking server is prepared to believe that WB is working on Bob's behalf, *provided* that its trusted password server (PS) verifies the pair $\langle Bob, pwd \rangle$. Thus, the banking server is governed by the following rule:

$$((WB \wedge PS) \text{ says } (\langle Bob, pwd \rangle \text{ speaks\_for } Bob)) \supset (WB|Bob) \text{ speaks\_for } (S|Bob) \quad (3)$$

Notice that, implicitly, the banking server believes that the only way WB would have Bob's pwd is if Bob provided it to allow WB to make requests on Bob's behalf.

The server passes the username-password pair on to the password server, which confirms the validity of that pair. Thus, from the perspective of the online-banking server,

$$PS \text{ says } (\langle Bob, pwd \rangle \text{ speaks\_for } Bob) \quad (4)$$

Combining the pieces of information (2) and (4), S determines that

$$(WB \wedge PS) \text{ says } (\langle Bob, pwd \rangle \text{ speaks\_for } Bob).$$

Therefore, using Rule (3), the banking server deduces that

$$WB|Bob \text{ speaks\_for } S|Bob.$$

Because speaks_for is a monotonic operator, this fact yields

$$WB|Bob \text{ speaks\_for } (WB|Bob \wedge S|Bob),$$

and hence

$$(WB|Bob) \text{ speaks\_for } (WB \text{ for}_S \ Bob).$$

Combining this result with the original request (1), the banking server can deduce that

$$WB \text{ for}_S \ Bob \text{ says } r.$$

In accordance with its access-control policy, the online banking server then grants the request.

We return to this example in Section 6, where we discuss how it can be verified in higher-order logic and HOL. However, we first describe how the logic itself can be embedded into HOL.

5 Implementing the Logic in HOL

In the previous sections, we have described the logic for access control and showed how it can be used to reason about brokered requests. Furthermore, this logic played an important role in the development of the CSIv2 standard, by providing a way to consider how to interpret requests and the SAS data structure.

To provide mechanized support for reasoning about the standard and its applications, we have embedded a subset of the logic into the Cambridge HOL (Higher Order Logic) theorem prover [4]. HOL supports a predicate calculus that is typed and allows variables to range over predicates and functions. HOL is implemented using the meta-language ML [6] and, at the lowest level, is a collection of ML functions that operate on sequences. The benefits of using HOL (or any open theorem prover) are at least threefold: (1) our results are formally verified and checked, (2) our results are easily reused by others and can be independently checked, and (3) future modifications and extensions to HOL theories can be easily verified for correctness.

There are multiple ways to embed a logic in HOL. The simplest way is to perform an *axiomatic extension* of the HOL logic: all of the desired theorems of the embedded logic (e.g., the axioms and inference rules in Figure 1) are defined as unproved axioms in HOL. This method is easy, but it may result in an inconsistent theory, thereby defeating the point of theorem proving.

The other possibility is to perform a *conservative extension* of the HOL logic: the desired logic is introduced as a series of definitional extensions in HOL, and all desired axioms and theorems are proved within the HOL system. Although this method requires much more work than an axiomatic extension, any resulting theories are guaranteed to be sound.

We therefore choose this latter approach, which requires the following steps:

1. Creating HOL datatypes for principals, formulas, and Kripke structures, along with their type constructors
2. Defining in HOL the models operator ($\models$)
3. Proving the soundness of the desired axioms and inference rules

We consider these steps in turn, as we give a flavor of the embedding.

The first step is to introduce new HOL datatypes for principal expressions, logical formulas, and so on. To do this, we use the `Hol_datatype` command, as in the following example:

```
Hol_datatype principal = name of string
                       | meet of principal →  principal
                       | quoting of principal →  principal
                       | forp of principal →  principal →  principal;
```

Recall that, for any structure $\mathcal{M} = \langle W, w_0, I, J \rangle$ and world $w \in W$, the property $\mathcal{M}, w \models \varphi$ is true if and only if $w \in \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$. Therefore, one way of defining the models predicate in HOL is first to define the meaning function $\mathcal{E}_{\mathcal{M}}[\![-]\!]$,

which in turn requires a definition for the extended interpretation function $\tilde{J}$. We have defined these functions in HOL as follows:

```
val JextDef =
    ⊢def  (Jext (World,w0,Ifn,Jfn) (name A) = Jfn World (name A)) ∧
      (Jext (World,w0,Ifn,Jfn) (P meet Q) =
       Jext (World,w0,Ifn,Jfn) P UNION Jext (World,w0,Ifn,Jfn) Q) ∧
      (Jext (World,w0,Ifn,Jfn) (P quoting Q) =
       Jext (World,w0,Ifn,Jfn) Q O Jext (World,w0,Ifn,Jfn) P) ∧
      (Jext (World,w0,Ifn,Jfn) (P forp D) Q =
       (Jext (World,w0,Ifn,Jfn) Q O Jext (World,w0,Ifn,Jfn) P) UNION
       (Jext (World,w0,Ifn,Jfn) Q O Jext (World,w0,Ifn,Jfn) D))  : thm

val EfnDef =
    ⊢def  (Efn (World,w0,Ifn,Jfn) (pv s) = Ifn World (pv s) INTER World) ∧
      (Efn (World,w0,Ifn,Jfn) (nots s1) =
       World DIFF Efn (World,w0,Ifn,Jfn) s1) ∧
      (Efn (World,w0,Ifn,Jfn) (s1 ands s2) =
       Efn (World,w0,Ifn,Jfn) s1 INTER Efn (World,w0,Ifn,Jfn) s2) ∧
      (Efn (World,w0,Ifn,Jfn) (s1 ors s2) =
       Efn (World,w0,Ifn,Jfn) s1 UNION Efn (World,w0,Ifn,Jfn) s2) ∧
      (Efn (World,w0,Ifn,Jfn) (s1 imps s2) =
       World DIFF Efn (World,w0,Ifn,Jfn) s1 UNION
       Efn (World,w0,Ifn,Jfn) s2) ∧
      (Efn (World,w0,Ifn,Jfn) (s1 eqs s2) =
       (World DIFF Efn (World,w0,Ifn,Jfn) s1 UNION
        Efn (World,w0,Ifn,Jfn) s2) INTER
       (World DIFF Efn (World,w0,Ifn,Jfn) s2 UNION
        Efn (World,w0,Ifn,Jfn) s1)) ∧
      (Efn (World,w0,Ifn,Jfn) (P says s1) =
       { w14 | Rfn (World,w0,Ifn,Jfn) P Rwith w14 SUBSET
        Efn (World,w0,Ifn,Jfn) s1}) ∧
      (Efn (World,w0,Ifn,Jfn) (P speaks_for Q) =
       { y | Rfn (World,w0,Ifn,Jfn) Q SUBSET Rfn (World,w0,Ifn,Jfn) P ∧
        y IN World})
      (Efn (World,w0,Ifn,Jfn) (P eqp Q) =
       {y | (Jext (World,w0,Ifn,Jfn) P = Jext (World,w0,Ifn,Jfn) Q) ∧
            y IN World}) : thm
```

We can then formally define the models predicate ($\models$) as follows:

```
val ModelFnDef =
    ⊢def  ∀World w0 Ifn Jfn w1 s.
        ((World,w0,Ifn,Jfn),w1) MD s =
        w1 IN World ⊃  w1 IN Efn (World,w0,Ifn,Jfn) s : thm
```

Using the models predicate, we proved the soundness of the axioms introduced in [5]. Specifically we proved as sound the standard modal properties (S2-S4). We did not directly prove sound S1, which corresponds to the first rule in Figure 1:

$$\frac{}{\vdash \varphi} \quad \text{if } \varphi \text{ is an instance of a propositional-logic tautology}$$

To prove this in HOL would require proving each tautology separately. Instead, we proved only those tautologies that we needed for our specific examples.

We also proved as sound the axioms relating the modal logic to the calculus of principals (i.e., the axioms P1, P2, P3, P5, P6, P7). However, we did not prove soundness of their hand-off axiom (P10), because (as noted in their paper) it is not sound. Appendix A provides a partial list of axioms and theorems we have proved sound.

6 Banking Example in HOL Theorem Prover

Having described our HOL implementation, we can now return to the online-banking example introduced in Section 4. In that scenario, an online-banking server (*SD*) receives a request from a web browser (*WB*) purportedly working as a broker on behalf of a bank client Bob. Determining whether or not that request should be honored required a chain of reasoning based on statements regarding who spoke for whom, delegation relationships, and requests. That chain of reasoning is captured by the following theorem:

```
∀WB Bob BobPwd PS SD req World w0 Ifn Jfn w1.
    ((World,w0,Ifn,Jfn),w1) MD
        (WB says BobPwd speaks_for Bob) imps
        (PS says BobPwd speaks_for Bob) imps
        (((WB meet PS) says BobPwd speaks_for Bob) imps
         (WB quoting Bob) speaks_for (SD quoting Bob)) imps
        (WB says Bob says req) imps
        ((WB forp SD) Bob says req)
```

This goal can be proved by applying a series of HOL tactics to the logical rules given in Figure 1. As mentioned previously, we cannot use the first rule of Figure 1 directly. Instead, we must introduce particular instances of this rule, which we can then verify in HOL. To prove the desired theorem, we need four specific instances, one of which is shown below:

$$\begin{aligned}\vdash &(\varphi_1 \text{ imps } (\varphi_2 \text{ imps } (\varphi_3 \text{ imps } (\varphi_4 \text{ imps } \varphi_5)))) \text{ imps}\\ &((\varphi_5 \text{ imps } (\varphi_4 \text{ imps } \varphi_6)) \text{ imps}\\ &(\varphi_1 \text{ imps } (\varphi_2 \text{ imps } (\varphi_3 \text{ imps } (\varphi_4 \text{ imps } \varphi_6)))))\end{aligned}$$

To verify the soundness of these new rules in HOL system, we must show that they are satisfied by all Kripke models. Thus, for example, we must prove the following theorem in HOL:

```
⊢ ∀s1 s2 s3 s4 s5 s6 World w0 Ifn Jfn w1.
    ((World,w0,Ifn,Jfn),w1) MD
          ((s1 imps (s2 imps (s3 imps (s4 imps s5)))) imps
          ((s5 imps (s4 imps s6)) imps
           (s1 imps (s2 imps (s3 imps (s4 imps s6))))) : thm
```

Proving the soundness of these rules is time consuming but straightforward. However, having to prove them significantly increases the burden for proving the desired theorem about the banking scenario.

7 Conclusions

Our original objective was to implement an access-control logic in HOL in order to reason about the CSIv2 protocol in ways that are assured and independently verifiable using mechanical theorem provers. This objective has been met by the current implementation. We expect that it would be straightforward for users of theorem provers other than HOL to inspect our definitional extensions and replicate the axioms, soundness proofs, theorems, and examples in the theorem prover of their choice.

In the course of doing our proofs, we investigated proving a soundness theorem directly:

> If statement φ is derivable from the access-control axioms, then φ is satisfied by all Kripke models.

The advantage of having such a theorem would be that users could do all their reasoning at the level of the access-control logic—instead of at the level of Kripke structures—and know that their conclusions were sound. In addition, such theorem would support using different models of the access-control logic (e.g., the MSS model) without affecting the proofs done by users: the sole requirement would be that the appropriate soundness theorem would have to be proved beforehand. Unfortunately, the prospects of introducing a derivability predicate and then proving soundness in HOL did not look promising, due to the dependence based on propositional-logic tautologies. Logical frameworks such as Isabelle/HOL [7] might better facilitate such an approach.

As stated previously, the access-control logic we have implemented was used to give a formal interpretation of part of the CSIv2 standard. As the CSIv2 standard is extended to provide a standard interpretation for the security-attributes component of the SAS data structure, there will be a need to reason about authorizations and privileges. We anticipate extending the current logic to support such reasoning. The benefits of such an extension include the ability to give a precise and accurate interpretation of the standard and a means for formal assurance of correctness and security.

8 Acknowledgments

This work was supported in part by the New York State Center for Advanced Technology in Computer Applications and Software Engineering (CASE).

References

1. CORBAServices: Common Object Services. Technical Report formal/98-07-05, Object Management Group, July 1998. Available via http://www.omg.org/cgi-bin/doc?formal/98-07-05.
2. The common secure interoperability version 2. Technical Report ptc/01-06-17, Object Management Group, June 2001. Available via http://www.omg.org/cgi-bin/doc?ptc/01-06-17.
3. Martin Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, September 1993.
4. M.J.C. Gordon and T.F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, New York, 1993.
5. Butler Lampson, Martin Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, November 1992.
6. Robin Milner, Mads Tofte, Robert Harper, and David MacQueen. *The Definition of Standard ML (Revised)*. MIT Press, 1997.
7. Lawrence C. Paulson. *Isabelle: A Generic Theorem Prover*. Springer, Lecture Notes in Computer Science 828, 1994.
8. Edward Wobber, Martin Abadi, Michael Burrows, and Butler Lampson. Authentication in the Taos operating system. *ACM Transactions on Computer Systems*, 12(1):3–32, February 1994.

A Theorems

We include a partial list of the axioms and derivable theorems of the authentication logic we have proved in HOL. The axioms included here correspond to those given in the appendix of [5].

val S2Thm =
⊢ ∀*s1 s2.*
(∀*World w0 Ifn Jfn w1.* ((*World,w0,Ifn,Jfn),w1) MD s1*) ∧
(∀*World w0 Ifn Jfn w1.* ((*World,w0,Ifn,Jfn),w1) MD* (*s1 imps s2*)) ⊃
∀*World w0 Ifn Jfn w1.* ((*World,w0,Ifn,Jfn),w1) MD s2 : thm*

val S3'2Thm =
⊢ ∀*World w0 Ifn Jfn w1 P s1 s2.*
((*World,w0,Ifn,Jfn),w1) MD*
(((*P says s1*) *ands P says s1 imps s2*) *imps P says s2*) *: thm*

val S4Thm =
⊢ ∀*s.*
(∀*World w0 Ifn Jfn w1.* ((*World,w0,Ifn,Jfn),w1) MD s*) ⊃
∀*P World w0 Ifn Jfn w1.* ((*World,w0,Ifn,Jfn),w1) MD* (*P says s*) *: thm*

```
val S5Thm =
    ⊢ ∀P s1 s2 World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD
      ((P says s1 ands s2) eqs (P says s1) ands P says s2) : thm

val P1Thm =
    ⊢ ∀P Q s World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD
      (((P meet Q) says s) eqs (P says s) ands Q says s) : thm

val P2Thm =
    ⊢ ∀P Q s World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD
      (((P quoting Q) says s) eqs P says Q says s) : thm

val P3Thm =
    ⊢ ∀P Q s World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD
      ((P eqp Q) imps (P says s) eqs Q says s) : thm

val P4AssocThm =
    ⊢ ∀P Q R World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD
      (((P meet Q) meet R) eqp (P meet Q meet R)) : thm

val P4CommuThm =
    ⊢ ∀P Q World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD ((P meet Q) eqp (Q meet P)) : thm

val P4IdemThm =
    ⊢ ∀P World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD ((P meet P) eqp P) : thm

val P5Thm =
    ⊢ ∀P Q R World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD
      (((P quoting Q) quoting R) eqp (P quoting Q quoting R)) : thm

val P6LeftThm =
    ⊢ ∀P Q R World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD
      ((P quoting Q meet R) eqp ((P quoting Q) meet P quoting R)) : thm

val P6RightThm =
    ⊢ ∀P Q R World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD
      (((Q meet R) quoting P) eqp ((Q quoting P) meet R quoting P)) : thm

val P7Thm =
    ⊢ ∀P Q World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD
      ((P speaks_for Q) eqs P eqp (P meet Q)) : thm

val P8Thm =
    ⊢ ∀P Q s World w0 Ifn Jfn w1.
      ((World,w0,Ifn,Jfn),w1) MD
      ((P speaks_for Q) imps (P says s) imps Q says s) : thm
```

```
val P9Thm =
    ⊢ ∀P Q World w0 Ifn Jfn w1.
        ((World,w0,Ifn,Jfn),w1) MD
        ((P eqp Q) eqs (P speaks'for Q) ands Q speaks'for P) : thm

val P12Thm =
    ⊢ ∀P Q P' World w0 Ifn Jfn w1.
        ((World,w0,Ifn,Jfn),w1) MD
        ((((P' meet Q) speaks'for P) ands Q speaks'for P') imps
         Q speaks'for P) : thm
```