A Rigorous Approach to Teaching Access Control

Shiu-Kai Chin and Susan Older
Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, New York 13244
http://www.ecs.syr.edu/faculty/chin http://www.cis.syr.edu/∼sueo

***Abstract*— Access control is central to any study of security. Verifying that access control is correctly implemented is essential but difficult, because meaningful verification requires formal derivations and proofs. We present an approach to teaching access control using an access-control logic based on modal logic. Students use the logic to formally describe and analyze a variety of access-control scenarios. The access-control logic plays a role analogous to that played by propositional logic in hardware design.**

We have developed a short course consisting of detailed presentations, examples, in-class exercises, homework projects of increasing complexity, and grading rubrics. These components were developed with detailed educational outcomes and outcomes-based assessments in mind. We have taught this short course in one 8-hour day as part of the Advanced Course in Engineering in Cyber Security to 49 rising juniors and seniors drawn from universities across the US. We present the design details of our short course, our experience, and lessons learned.

***Index Terms*— Access control, formal methods, modal logic, educational outcomes, outcomes assessment.**

I. Introduction

Access-control policies and the system components that implement those policies are central to any study of computer, network, or information security. Unlike other computer engineering topics, however, access control is typically taught with little in the way of mathematically rigorous analytical techniques. As a result, students are bereft of the means to precisely specify, describe, and verify how access to resources is protected. In contrast, propositional logic and finite-state machine theory is routinely taught at the freshman and sophomore levels in digital-design courses, providing a rigorous foundation for all of computer hardware design. Consequently, students have the ability to formally specify and verify their hardware designs.

The capability to formally specify and verify designs is particularly crucial for subsystems whose task is to guard access to resources. These guards, which we call *reference monitors*, are responsible for mediating all access requests to protected resources such as files, medical information, and physical spaces. All reference monitors must satisfy three essential properties: (1) *completeness*: the reference monitor cannot be bypassed, (2) *isolation*: the reference monitor is tamper proof, and (3) *verifiability*: the reference monitor is known to be correctly implemented. Without means to formally specify and verify reference monitors, this third property is unattainable.

Our goal is to achieve the same level of rigor when teaching access control as is achieved in hardware curricula. Our view is this: if you are the hardware designer and you are given the input values to your design, then you should be able to justify mathematically whether the value on any particular output is a 0 or a 1. Similarly, if you are the security engineer who has security requirements to meet and you are given a policy and a request, you should be able to justify mathematically if your answer is a "yes" or a "no."

What we describe in this paper is how we organized and conducted a one-day 8-hour intensive short course using formal logic to teach access control. We taught this course as part the Advanced Course in Engineering (ACE) Cyber Security Boot Camp ([1], [2]), a 10-week intensive summer course on computer and network security combining work, education, and leadership-development activities. The most notable feature of the Advanced Course is the focus on delivering an extensive engineering report detailing a solution to an engineering problem of realistic complexity. These reports typically take 20 to 30 hours of work each week in addition to the 8 hours in class, 3 days in internships, and a day of leadership activities. Students work in teams of three. Penalties for late reports are severe: one late report results in no credit for the week. Two late reports result in dismissal from the program.

The 2005 class consisted of 49 students drawn from 40 universities across the US. Of the 49 students, 35 were ROTC students, 6 were part of the National Science Foundation Scholarship for Service program, and the 8 remaining students were from other programs. The backgrounds of the students included engineering and computer science as well as a few non-engineering disciplines. Due to the relatively large number of students, the class was split into two sessions: one session had lectures every Monday, while the other session had lectures every Tuesday.

Formal methods are notoriously difficult to teach and arguably more difficult to convey in ways that provide students useful tools. While we were confident that the students were highly motivated and willing to work hard, we had several questions regarding what we could realistically expect of them after a single 8-hour day:

1) In the focused area of justifying access control decisions based on reference monitors, could we get rising juniors and seniors to use formal methods?
2) Could the students do real derivations that would be respected by logicians and mathematicians?
3) Could we impart enough knowledge in access control and formal methods in a single 8-hour day to enable students to solve access-control problems formally?

The answer to all of the above questions is a qualified "yes."

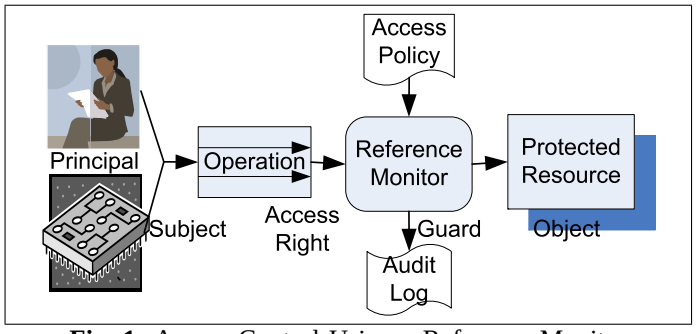


Fig. 1: Access Control Using a Reference Monitor

We were successful with those students from engineering, computer science, and mathematics programs; we were unsuccessful with students from other backgrounds. This paper describes how we designed and delivered the lectures, as well as our assessment results.

In Section II, we provide some background on our approach to viewing and teaching access control. Because the structure of our course emerged from our desired educational outcomes, we present our educational outcomes in Section III and our lecture structure in Section IV. Our success depended crucially upon the in-class exercises woven through the day: we describe them in Section V. We present the results of the in-class exercises in Section VI and our lessons learned in Section VII. This section also describes modifications we made based on our experiences. We offer conclusions and final remarks in Section VIII.

II. Background

Figure 1 provides an abstract view of how a reference monitor fits into a system. Principals—either people, machines, or processes—make requests to perform some operation on a protected object. Principals that make requests are *subjects*, and permission to perform an operation on an object is called an *access right* or *privilege*. The reference monitor's job is to decide whether to say "yes" or "no." The decision depends on several factors: what is being asked for, potentially the identity and other attributes of the subject making the request, and finally the access policy. A record of all the access requests and decisions is kept in the audit log.

Our objective is to enable engineers to (1) formally describe principals, requests, and policies, (2) explicitly state any trust assumptions, and (3) formally prove the correctness of access-control decisions. Specifically, we want to equip engineers with the mathematical means to analyze access-control situations and to justify access decisions in a variety of situations: access to physical resources such as movie theaters; access to information resources, such as files; and access to computer hardware resources, such as physical memory segments. These decisions require specific and precise descriptions of requests, policies, trusted principals, and jurisdiction of authority. Principals need to be authenticated and authorized using mechanisms such as tickets, access-control lists, and capabilities. We intend that hardware, software, and systems engineers use the mathematical methods described in this paper to describe and verify their designs.

Lampson provides a glimpse of this vision in his high-level overview [3], where he cites access control as being crucial to security. He focuses on questions related to local and distributed access control, chains of trust as a means to associate requests and statements with principals, jurisdiction of authority, and verifying delegation. His explanations use an access-control logic developed with Abadi, Burrows, and Wobber [4], later extended by Howell and Kotz [5]. This access-control logic—used but undefined in [3]—has a semantics based on Kripke structures. Several versions of the logic appear in [4], [6], and [7]. The versions differ in their choice of axioms and their use of syntactic sugar.

As part of our collaborative research and teaching efforts, we have identified a small set of sound inference rules that are nonetheless sufficient for defining a variety of access-control situations. Our experience is that a small set of inference rules—easily expanded by the addition of derived rules—makes learning to do derivations and proofs simpler for our students.

In planning the 8-hour intensive short course, we realized that we needed clarity on the specific outcomes we hoped to achieve and that every element of the day had to contribute to the desired outcomes. Toward this end, we did the following:

1) Described the incoming knowledge, skills, and abilities we expected of students.
2) Described educational outcomes in detail (i.e., described what students are able to do if they successfully complete the short course).
3) Devised a sequence of short course components (lectures, in-class exercises, and problems), with interlocking educational outcomes and assessments, that in combination could produce the desired results.

III. Required and Acquired Knowledge

In this section, we describe in detail the required and acquired knowledge expected of our students. First, however, we provide a short description of our methods for describing knowledge and outcomes. Our experience is that using these methods, which focus on observable student behavior, provide clarity and precision when thinking about educational objectives and when assessing student learning.

A. Describing Knowledge and Outcomes

When describing knowledge, skills, and abilities, we have found it beneficial to employ a disciplined and precise approach to describing knowledge, skills, and abilities. Instead of writing descriptions using words that are open to many interpretations (e.g., *to know, to understand, to appreciate*), our approach is to produce objectives containing two or three basic elements as described by Diamond in [8] (page 132):

- a verb that describes an observable action,
- a description of the condition under which the action takes place: "when given *x*, you will be able to . . .", and
- the acceptable performance level—that is, what percentage of correct answers will be considered

TABLE I
KNOWLEDGE LEVEL AND ASSOCIATED VERBS

Knowledge Level	Associated Verbs
Recall	define, repeat, record, list, name, retrieve
Comprehension	restate, discuss, describe, recognize, explain, identify, locate, report, review, tell, translate, interpret, extrapolate
Application	use, demonstrate, sketch, schedule, practice, illustrate, operate
Analysis	distinguish, differentiate, calculate, experiment, test, compare, contrast, criticize, diagram, inspect, debate, solve
Synthesis	construct, organize, compose, plan, manage, design, formulate, arrange, assemble, prepare, set up, create
Evaluation	measure, score, estimate, choose, assess, judge, appraise, evaluate, rate

To do well in this course, you must be familiar and comfortable with propositional logic, sets, and relations. You must be able to construct rigorous arguments/proofs about these topics and use induction where warranted. More specifically, the knowledge required of propositional logic at various levels are:

Comprehension

C1: When given a formula in propositional logic and its interpretation, you should be able to explain its meaning in English.

C2: When given a precise but informal description, you should be able to represent the description symbolically and give its interpretation.

Analysis

An1: When given a set of assumptions and a goal to prove, you should be able to prove or disprove the goal, using the rules of inference to calculate whether the goal is true.

An2: When given definitions of operations on sets or relations, you should be able to calculate the result of applying those operations to specific sets.

Synthesis

S1: When given a set of assumptions and a (true) goal to prove within an underlying theory, you should be able devise a proof strategy to prove the goal based on the form of the goal, definitions, and theorems of the underlying theory.

Evaluation

E1: When given a theory, inference rules, and a proof, you should be able to judge if the proof is correct.

Fig. 2: Assumed Incoming Skills and Abilities

> acceptable, how many errors will be permitted, how many and which examples must be included, and so on.

Our descriptions use Bloom's taxonomy of knowledge [9]. We use the approach described above in conjunction with observable actions associated with Bloom's different levels of knowledge, as shown in Table I.

B. Required Knowledge

Figure 2 describes the knowledge, skills, and abilities students ideally should have before taking our course. Essentially, we are expecting them to have an undergraduate knowledge of propositional logic and set theory as found in most undergraduate treatments of discrete mathematics.

Because students come with varying degrees of preparation, we had to plan for students with deficient mathematical backgrounds. Students worked in teams of three, organized so that each team had at least one computer science or engineering student. We will comment more on this later.

C. Acquired Knowledge

Figure 3 describes what we hoped students would learn in the short course. The items listed under *Comprehension* identify the major topics. This list is similar to the coverage found in standard computer and network security courses. As our course is reference-monitor centric, we focused on tickets, access-control lists, and capabilities as mechanisms for controlling access. Our discussion of cryptography centered on its use as a means to authenticate and identify statements and principals as opposed to a means for preserving privacy.

The remainder of the outcomes detail what we believe to be the unique aspects of our short course.

- **Application:** These outcomes relate to how we hope students will be able to use the access-control logic as a policy-specification and implementation-description language.
- **Analysis:** These outcomes describe in detail what kind of derivations and calculations we expect students to do. These goals are analogous to what is expected of students in digital-design courses.
- **Synthesis:** These outcomes attempt to address the more subtle aspects of security, such as "what are the underlying assumptions regarding who is trusted on what?" and on the evidence used in "real life" situations.
- **Evaluation:** This outcome attempts to get students to assume the role of evaluator, certifier, or critic in a rigorous and precise way.

The linchpin of our day are the *analysis* outcomes. In particular, we have found over the years that making the link from expressions in the access-control logic to the underlying Kripke semantics is crucial for avoiding hours of needless debate over expressions such as the following:

$$K_{Alice} \text{ says } \text{“Read file foo”}$$
$$\text{Alice controls } \text{“Read file foo”}$$

Students without a means to calculate precisely the mathematical meaning of expressions ultimately are left to purely intuitive (and usually inaccurate) notions of what an expression *could* mean. Mere symbol pushing is inadequate. Much of our day was organized around realizing the *analysis* outcomes and providing enough context and practice in class so students had a reasonable chance at incorporating the access-control logic into their thinking.

IV. PLANNING THE DAY

Our major concern in planning the day was to ensure the active participation of students. Specifically, we wanted students to practice using the logic and doing proofs using the

Comprehension

C1: Define the meaning of security services such as confidentiality, integrity, non-repudiation, and authentication.
C2: Describe the characteristics of private-key and secret-key cryptographic systems.
C3: Describe basic principles of trust topologies and networks of certification authorities.
C4: Describe and define the concept of a reference monitor.
C5: Describe how tickets and capabilities are used for access control.
C6: Describe how access-control lists are used for access control.
C7: Describe how address descriptors and protection descriptors are used for process isolation, memory segmentation, and ticket-based access control.

Application

Ap1: When given protocol descriptions and trust hierarchies, you should be able to use the logic of authentication for distributed systems to describe the protocol and trust relationships.
Ap2: When given a trust topology, you should be able to determine the necessary certificates for establishing trust in a key.
Ap3: When given a diagram and description of a memory protection scheme using address and protection descriptors you should be able to write down formulas in the access-control logic describing the interpretation of requests, tickets, descriptors, and protection policies enforced by the reference monitor.

Analysis

An1: When given a set of assumptions and a security goal to prove, you should be able to prove, using formal inference rules, if the security goal is true or not.
An2: When given an informal description of users, roles, and privileges, you should be able to analyze security properties such as role membership, mutual exclusion, etc.
An3: When given a Kripke structure, you should be able to determine the beliefs of principals (simple and compound) and whether or not one principal speaks for another.

Analysis, continued

An4: When given an axiom or theorem in the access-control logic, you should be able to prove its soundness in the underlying Kripke model.
An5: When given a set of certificates, you should be able to formally derive whether a key is associated with a particular principal.
An6: When given formulas in the access-control logic describing the interpretation of requests, tickets, descriptors, and protection policies enforced by the reference monitor, for any given request you should be able to infer using the inference rules of the access-control logic if the request is granted or denied.
An7: When attempting a formal analysis of a reference monitor, deduce the underlying trust assumptions being made.

Synthesis

S1: When given a "real life" scenario that incorporates subjects, objects, permissions, certificates, and descriptors, you should be able to formally describe the scenario in the access-control logic and show how access-control decisions are made using the inference rules of the access-control logic.
S2: When given a description of a trust topology, you should be able to create a formal description of the certificates and trust relationships for the certification authorities.
S3: When given a layered descriptor-based protection scheme, formalize the policies, interpretations of tickets and access-control lists, requests, and trust assumptions showing how integrity is maintained.
S4: When given an informal description of a protection policy, be able to create a formal description that highlights the interaction between requests, tickets, access-control lists, policies, and *trust assumptions*.

Evaluation

E1: When given a theory, inference rules, and a proof, you should be able to judge if the proof is correct.

Fig. 3: Educational Outcomes

inference rules *in class* so that questions could be answered while we were in the room and so that we could spot and correct mistakes in their thinking.

Table II presents our agenda for the day, to which we pretty faithfully adhered; we ended up adding an extra 30 minutes to the schedule due to small time overruns. The time allocation reflects our view that theory and practice are of equal value. We thought it absolutely essential that students successfully make it through the theory portion of class before lunch: they needed to be able to write well-formed expressions, be able to calculate the meaning of expressions (which we hoped would help them describe the informal scenarios in the logic later), and use the inference rules to develop formal proofs. Lunch would provide an opportunity for them to digest the logic before moving on to applications in the afternoon.

We carefully crafted the sequence of topics so that each element would build upon the previous ones.[1] The schedule also reflects our intended assessment structure. We had at least one exercise keyed to each of the main topics in the schedule. We were prepared to slow down (and drop later topics if necessary) to re-cover material if students were unable to do a particular exercise.

[1]In actuality, the cryptography and keys portion of the morning were not central to our work. However, we had promised to cover those topics for other instructors to build on in subsequent lectures during the summer. By introducing these topics early, we not only satisfied our obligation but also could use them as a source of examples for the remainder of the day.

TABLE II
AGENDA FOR 8-HOUR SESSION

Time	Topic	Presenter
9:00–9:10	Introduction	Chin
9:10–10:00	Cryptography and Keys Secure Access Control	Chin
10:00–10:20	Discrete Mathematics	Older
10:20–10:45	Propositional Logic	Older
10:45–11:00	Break	
11:00–12:45	Access-Control Logic	Older
12:45–1:45	Lunch	
1:45–3:10	Simple & Basic Access-Control Concepts	Chin
3:10–3:25	Break	
3:25–4:30	Isolation and Sharing	Chin
4:40–5:00	Problem Description	Chin & Older

V. IN-CLASS EXERCISES

As part of the ACE, students were separated into *flights* of three students each (one flight had four) for the duration of the summer. These flights had been constructed to ensure that each flight had at least one computer engineering or computer science student. We used these flights as the basis for our in-class group exercises.

Each major topic area of the agenda included one or more ten-minute group exercises to solve in class. All flights worked on each exercise. After each exercise, we selected one flight at random (via index cards) to present their solutions, and we

TABLE III
IN-CLASS GRADING CRITERIA

Outcome	Points
Nice try but incorrect	1
Some parts correct but still some major errors or omissions	3
Largely correct	5

1) Let $X = \{x, y, z\}$ and $W = \{t, w, x\}$. Calculate:
   - 1_X
   - $X \cap W$
   - $X \times W$
2) Consider the following relations:

$$R_1 = \{(x,y),(y,w),(y,t),(t,x),(t,w)\}$$
$$R_2 = \{(x,y),(y,z),(z,t),(z,x),(z,w)\}$$

   Calculate the following:
   - $R_2(z)$
   - $R_2(t)$
   - $R_1 \circ R_2$
   - $R_1 \circ R_1$

Fig. 4: Discrete Mathematics Group Exercise

asked the other flights to critique the proposed solutions. We awarded points (as part of a summer-long competition among the flights) based on the rubric given in Table III.

We describe the exercises in this section; the assessment results appear in Section VI.

A. Discrete Mathematics

We designed the discrete-mathematics portion of the schedule to review core concepts (sets and relations) that lie at the heart of the semantics of the access-control logic. We also wanted to establish a common notation for the duration of the lecture, particularly for concepts (e.g., identity relations, composition, and relational images) that the literature tends to denote in multiple ways. We viewed this topic as prerequisite knowledge that the students should have seen before, but we recognized the need to explicitly review the concepts before using them in earnest.

The group exercise, shown in Figure 4, was designed to ensure that students could apply basic set operations (incoming ability **An2** from Figure 2), such as intersection ($\cap$), cross products ($\times$), and relational composition. The exercise made use of the following notation and conventions (where A is a set and R, S are binary relations over A):

- **Identity relation on a set** A**:** $1_A = \{(a,a) \mid a \in A\}$
- **Images under a relation** R**:** For any $a \in A$, $R(a) = \{b \mid (a,b) \in R\}$.
- **Composition of relations** R **and** S**:** $R \circ S = \{(x,z) \mid \exists y((x,y) \in R \;\&\; (y,z) \in S)\}$

B. Propositional Logic

In a similar vein, we designed the propositional-logic review to remind students of the basic propositional-logic operators and how to use truth tables. Specifically, we wanted students to work with a familiar mechanism (i.e., truth tables) for providing interpretations of formulas and for determining the validity of formulas in propositional logic. When we later introduced Kripke-style semantics for the access-control logic, we wanted to be able to appeal to students' understanding of truth tables as interpretations.

- Translate the following sentences into propositional logic, using the following propositions:

  $J \equiv$ "Dr. Jabbour is happy with my progress"
  $R \equiv$ "I ran 8 miles today"
  $S \equiv$ "I overslept"

  1) Either I ran 8 miles today or Dr. Jabbour is not happy with my progress.
  2) If I overslept, then I did not run 8 miles today.
  3) If I did not oversleep, then Dr. Jabbour is happy with my progress.
- Assume the first two statements are true. Is the third necessarily true? (Use a truth table to support your answer.)

Fig. 5: Propositional Logic Group Exercise

The group exercise, which appears in Figure 5, required students to first translate an ACE-related word problem into propositional logic and then to use truth tables to determine whether a specific argument was valid. In terms of Figure 2's incoming skills and abilities, these problems tested the incoming abilities **C2** and **An1**.

C. Access-control Logic

We spent roughly two hours introducing the access-control logic, starting with the syntax of well-formed formulas. Based on prior experiences teaching this material, we believed it essential to have students work with the formal semantics (i.e., Kripke structures) of the logic before moving on to the formal inference rules. We introduced the inference rules and formal proofs only after students demonstrated adequate facility with the semantics.

To provide some context for the group exercises, we first provide a very brief overview of the syntax, semantics, and inference rules of the logic; more details appear in [10], [11].

1) Overview of the logic: The abstract syntax of logical formulas (ranged over by φ) is defined as follows, where P and Q range over a countable set of *principal names* and p ranges over a countable set of *propositional variables*:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \supset \varphi_2 \mid P \Rightarrow Q \mid P \text{ says } \varphi \mid P \text{ controls } \varphi$$

Informally, a formula $P \Rightarrow Q$ (pronounced "P speaks for Q") indicates that every statement made by P can also be viewed as a statement from Q. A formula P controls φ is syntactic sugar for the implication $(P \text{ says } \varphi) \supset \varphi$: in effect, P is a trusted authority with respect to the statement φ.

The semantics of formulas is given via Kripke structures, as given by the following definitions.

Definition: A *Kripke structure* $\mathcal{M}$ is a three-tuple $\langle W, I, J\rangle$, where:

- W is a set, whose elements are called *worlds*.
- I : **PropVar** $\rightarrow \mathcal{P}(W)$ is an *interpretation* function that maps each propositional variable p to a set of worlds.
- J : **PName** $\rightarrow \mathcal{P}(W \times W)$ is a function that maps each principal name A into a relation on worlds (i.e., a subset of $W \times W$).

$\diamond$

Definition: Each Kripke structure $\mathcal{M} = \langle W, I, J\rangle$ gives rise to a function

$$\mathcal{E}_{\mathcal{M}}[\![-]\!] : \textbf{Form} \rightarrow \mathcal{P}(W),$$

where $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$ is the set of worlds in which φ is considered true. $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$ is defined inductively on the structure of φ, as follows:

$$\begin{aligned}
\mathcal{E}_{\mathcal{M}}[\![p]\!] &= I(p) \\
\mathcal{E}_{\mathcal{M}}[\![\neg\varphi]\!] &= W - \mathcal{E}_{\mathcal{M}}[\![\varphi]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \wedge \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \vee \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \supset \varphi_2]\!] &= (W - \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!]) \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\\
\mathcal{E}_{\mathcal{M}}[\![P \Rightarrow Q]\!] &= \begin{cases} W, & \text{if } J(Q) \subseteq J(P) \\ \emptyset, & \text{otherwise} \end{cases} \\
\mathcal{E}_{\mathcal{M}}[\![P \text{ says } \varphi]\!] &= \{w | J(P)(w) \subseteq \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]\} \\
\mathcal{E}_{\mathcal{M}}[\![P \text{ controls } \varphi]\!] &= \mathcal{E}_{\mathcal{M}}[\![(P \text{ says } \varphi) \supset \varphi]\!]
\end{aligned}$$

Note that, in the definition of $\mathcal{E}_{\mathcal{M}}[\![P \text{ says } \varphi]\!]$, $J(P)(w)$ is simply the image of world w under the relation $J(P)$. $\diamond$

The semantic functions $\mathcal{E}_{\mathcal{M}}$ provide a fully defined and fully disclosed interpretation for the formulas of the logic. We believe it is essential for students to work directly with the Kripke structures so that they realize the formulas have precise meanings not based purely on vague ideas about the English language. However, the true value of the logic as a means to reason about access-control situations emerges from a small core collection of sound inference rules and syntactic-sugar definitions (see Figure 6), along with a larger set of rules that can be formally derived from the core rules (see Figure 7 for the derived rules we gave the students). These rules become the basis for reasoning about access-control decisions. The Kripke structures are then only necessary if one wishes to add new inference rules and to verify their soundness.

2) Logic-related exercises: We had two group exercises during this session, and we allotted extra time for these exercises. The first exercise, displayed in Figure 8, required students to calculate the interpretations of particular formulas with respect to a specific Kripke structure (educational outcome **An3** from Figure 3).

The second exercise, displayed in Figure 9, required students to construct a formal proof of a specific goal, using only the given inference rules and derived rules discussed in lecture to that point (educational outcome **An1**).

Taut $\dfrac{}{\varphi}$ if φ is an instance of a prop-logic tautology

Modus Ponens $\dfrac{\varphi \quad \varphi \supset \varphi'}{\varphi'}$ *Says* $\dfrac{\varphi}{P \text{ says } \varphi}$

MP Says $\dfrac{}{(P \text{ says } (\varphi \supset \varphi')) \supset (P \text{ says } \varphi \supset P \text{ says } \varphi')}$

Speaks For $\dfrac{}{P \Rightarrow Q \supset (P \text{ says } \varphi \supset Q \text{ says } \varphi)}$

$P \text{ controls } \varphi \stackrel{\text{def}}{=} (P \text{ says } \varphi) \supset \varphi$

Fig. 6: Inference rules given to students

Conjunction $\dfrac{\varphi_1 \quad \varphi_2}{\varphi_1 \wedge \varphi_2}$ *Double negation* $\dfrac{\neg\neg\varphi}{\varphi}$

Simplification $\dfrac{\varphi_1 \wedge \varphi_2}{\varphi_1}$ *Simplification* $\dfrac{\varphi_1 \wedge \varphi_2}{\varphi_2}$

Disjunction $\dfrac{\varphi_1}{\varphi_1 \vee \varphi_2}$ *Disjunction* $\dfrac{\varphi_2}{\varphi_1 \vee \varphi_2}$

Modus Tollens $\dfrac{\varphi \supset \varphi' \quad \neg\varphi'}{\neg\varphi}$

Hypothetical Syllogism $\dfrac{\varphi_1 \supset \varphi_2 \quad \varphi_2 \supset \varphi_3}{\varphi_1 \supset \varphi_3}$

Disjunctive Syllogism $\dfrac{\varphi_1 \vee \varphi_2 \quad \neg\varphi_1}{\varphi_2}$

Controls $\dfrac{P \text{ controls } \varphi \quad P \text{ says } \varphi}{\varphi}$

Fig. 7: Derived rules given to students

Consider the Kripke structure $\mathcal{M}_1 = \langle W_1, I_1, J_1\rangle$, where:

$W_1 = \{w_0, w_1, w_2, w_3\}$
$I_1(q) = \{w_2, w_3\} \quad I_1(r) = \{w_0, w_2\} \quad I_1(t) = \{w_0, w_1, w_3\}$
$J_1(Cy) = \{(w_0, w_0), (w_0, w_2), (w_1, w_1), (w_2, w_2), (w_2, w_1), (w_3, w_0)\}$
$J_1(Di) = \{(w_0, w_1), (w_1, w_2), (w_2, w_3), (w_3, w_0), (w_3, w_1)\}$

Calculate the following:

- $\mathcal{E}_{\mathcal{M}_1}[\![(t \supset r) \vee (\neg q)]\!]$
- $\mathcal{E}_{\mathcal{M}_1}[\![Di \Rightarrow Cy]\!]$
- $\mathcal{E}_{\mathcal{M}_1}[\![Cy \text{ says } ((t \supset r) \vee (\neg q))]\!]$
- $\mathcal{E}_{\mathcal{M}_1}[\![Di \text{ says } (Cy \text{ says } ((t \supset r) \vee (\neg q)))]\!]$

Fig. 8: Kripke Structures Group Exercise

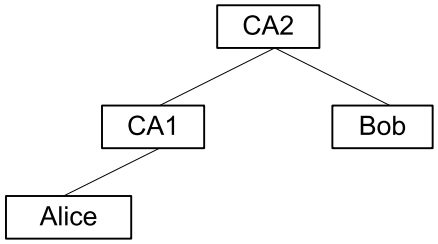
Consider the following collection of hypotheses:

1. *MC* controls (*Lee* controls $purchase$)
2. *CA* controls ($K_L \Rightarrow$ *Lee*)
3. $K_{CA} \Rightarrow$ *CA*
4. K_{CA} says ($K_L \Rightarrow$ *Lee*)
5. K_L says $purchase$
6. *MC* says (*Lee* controls $purchase$)

Give a formal proof of the statement $purchase$, using only the inference rules and derived rules discussed in class.

Fig. 9: Formal Proof Group Exercise

Consider the organization of certificate authorities below:



Alice wants to get Bob's public key and trusts that $K_{CA1} \Rightarrow CA1$. What certificates and assumptions about jurisdiction does Alice need to deduce that $K_{Bob} \Rightarrow Bob$. Do a formal proof.

Fig. 10: Trust Topology Group Exercise

The formal-proof exercise completed the first half of the lecture. We took an hour lunch break, to allow both students and lecturers to eat and relax.

D. Access-control Concepts

The afternoon's session began with a discussion of public-key infrastructures and how the relationship between principals and public-keys could be expressed in the logic. We also discussed certificates, certificate authorities, and trust topologies. As a group exercise (see Figure 10), we gave students a simple description of a trust topology and asked them to identify and formally describe the necessary certificates and trust relationships for certification authorities. This exercise addressed educational outcomes **An5** and **S2**.

We then moved on to core access-control concepts, including reference monitors, tickets and capabilities, and access-control lists. As group exercises, we asked students to apply the concepts to common everyday activities. In the context of tickets (see Figure 11), we asked students to describe how movie tickets work. Specifically, they had to represent in the logic the various components that underlie a ticket-collector's decision to allow a patron into a movie theater and to give the corresponding proof. For access-control lists (Figure 12), we asked students to describe a restaurant-reservations scenario. These exercises addressed educational outcome **S1**.

For the movie-theater ticket example, use the access-control logic and its inference rules to do the following:

1) Do a formal proof showing that a movie patron with a ticket should be let in to see the movie.
2) What additional assumption is needed to do the proof? What kind of assumption is this? What are the security implications regarding tickets?
3) If someone shows up without a ticket but makes a request to see the movie, will the ticket collector using the policy let him in?

Fig. 11: Tickets Group Exercise

For the dinner reservation example, use the access-control logic and its inference rules to do the following:

1) Formalize the maître d's reservation list.
2) Do a formal proof showing that Alice will be seated.
3) Is Alice's identity authenticated?
4) If someone shows up without a reservation and requests to be seated, will they be seated by the maître d'?

Fig. 12: ACLs Group Exercise

E. Isolation and Sharing

The final session of the day focused on process isolation and sharing. Specifically, we presented details on descriptors and memory management as described in the classic Saltzer and Schroeder paper [12] and expressed these concepts in the access-control logic.

We incorporated two group exercises into this session. The first (see Figure 13) required students to first translate a memory-access request using base-bound registers into the access-control logic and then deduce that the request should be honored. The second (Figure 14) required students to prove that a memory-write request with a write ticket should be honored. These exercises addressed educational outcomes **An1** and **An2**.

VI. Outcomes

As mentioned previously, students were divided into two sections (Monday's class and Tuesday's class) for the entire summer. We used the same agenda, slides, examples, and exercises for both days.

The assessment results for the group exercises appear in Table IV. In two instances (Monday's restaurant exercise and Tuesday's ticket exercise), two groups' scores are reported. In each case, we selected a second group to share their solution after the class identified problems with the first group's answer.

Consider the following request:

$$(256, 255) \text{ says } \langle 128, R \rangle$$

Given the memory access-control policy, prove:

$$\langle 128, R \rangle$$

Fig. 13: Memory-access Request Group Exercise

Give a formal proof for an address-write operation. Specifically,

$$(256, 255) \text{ says } \langle 128, W, 0\rangle$$

Assume the following:

1. $(256, 255)$ says $\langle 128, W, 0\rangle$	request
2. $write$ says $((128 \leq 255) \supset (256, 255)$ controls $\langle 128, W, 0\rangle)$	ticket
3. $Supervisor$ controls $((128 \leq 255) \supset (256, 255)$ controls $\langle 128, W, 0\rangle)$	policy
4. $write \Rightarrow Supervisor$	trust assumption

Fig. 14: Memory-write Request Group Exercise

Topic	Monday Group	Monday Score	Tuesday Group	Tuesday Score
Set Operations	Hotel	5	Juliet	1
Prop'l Logic	Foxtrot	5	Juliet	3
Kripke Structure	Alpha	5	Lima	5
Formal Proof	Delta	5	Kilo	5
PKI Proof	Alpha	3	Oscar	5
Tickets:	Alpha	3	Kilo	1
Box Office			November	5
ACLs:	Alpha	3	November	5
Restaurant	Delta	5		
Base-bounds	Charlie	5	Lima	5
Descriptors	Bravo	5	Lima	5

TABLE IV
ASSESSMENT RESULTS FOR GROUP EXERCISES

We were pleased and surprised by the students' ability to grasp the material in one day. We had started Monday's lecture fully expecting that we would be able to cover only a portion of our material. What we observed provided affirmative answers to our initial questions:

- By the end of the day, the students could competently use the rules of the access-control logic.
  In fact, by lunchtime, some students had already identified the utility of proving and then using derived inference rules to simplify subsequent problems. For example, Team Delta's solution to the formal-proof exercise included a proof for a derived rule that we had intended to introduce for use in the afternoon's session:
  $$\frac{P \Rightarrow Q \qquad P \text{ says } \varphi}{Q \text{ says } \varphi}$$
- Students could do some simple translations of English scenarios into the access-control logic.
- Students could analyze and solve simple access-control problems formally.

The result of the week-long problem set we assigned reaffirmed these observations. Students could use Kripke structures to calculate the meanings of formulas, translate between English and the access-control logic, do formal proofs, and analyze simple access-control problems. However, they were unable to take the next step of translating the memory-management mechanisms of Saltzer and Schroeder [12] into the logic. This result is not surprising, given the very intricate nature of this topic and their limited exposure to it.

VII. LESSONS LEARNED

Heading into the lecture, we were unsure whether we could even cover all the material in the allotted time. We designed the in-class group exercises to provide us with real-time feedback, so that we could adjust our pace and our expectations if necessary. These exercises were also vital for keeping students engaged in the lecture. We realized that students had begun to absorb the material when they began making jokes using the terminology of the logic by the afternoon. Several students commented positively (and unsolicited) at the ACE graduation two months later that the group exercises had been both a useful learning tool and an antidote to "death by powerpoint".

Based on our experience, we highly recommend having two instructors (instead of one) present if a boot-camp format is used. Teaching for eight hours is physically exhausting, and having the opportunity to recharge was essential. Furthermore, the presence of a second instructor was critical to our ability to stay to the agenda. Each of us watched the time as the other lectured, interrupting when the other strayed off topic.

However, the two-instructor approach also imposes its own demands. To pull this lecture off, we developed a very integrated and coordinated schedule. The high level of integration depended upon our ability to mesh our individual teaching styles and upon a level of trust built up over years of working together. Through our collaborative research in formal methods for security, we worked hard to come up with a logical system that was simple enough to cover in a short period of time and yet complete enough to express useful concepts and mechanisms. We were able to avoid some of the standard theory-practice wars because we both agreed upon the utility of educational outcomes. By focusing on the observable behaviors that we hoped to elicit from the students, we were able to agree on a detailed agenda and to stick to that schedule.

We found it helpful to build intuitive roads into the theory. Our belief is that everyone participates in access-control decisions on a daily basis, such as making dinner reservations, boarding planes, renting cars, or making online purchases. We were better able to motivate the logic and to get students working with it by appealing to their understanding of how the world works.

In retrospect, we believe the detailed discussion of isolation and sharing, while important, was a bridge too far. We used the logic to present details on hardware descriptors and memory management, moving well beyond the typical use of the classic Saltzer and Schroeder paper [12] to introduce the principle of least privilege. However, a half day's lecture is simply not sufficient time to adequately explicate all the details in a formal way, particularly when the formalism itself is new material.

Based on our experiences, we made several content changes

in our summer 2006 verson of the course. Students' evaluations of our 2005 lecture indicated that several students did not understand the connection between our lecture and the more hands-on lectures involving network attack and defense, steganography and information hiding, and wireless security. We therefore made more explicit connections between access control and the other network topics taught during the ACE. We also expanded the number of everyday access-control examples we used, so as to better build on students' intuition. Finally, we omitted the isolation and sharing topics, and replaced them with delegation, chains of trust, and proxies: these topics better supported the network focus of the ACE.

Demographically, all 2006 ACE students had electrical engineering, computer engineering, or computer science backgrounds, and hence they were better prepared for the mathematics and logic. The quality of the mathematics in the students' solutions and reports was much better than in previous years. More importantly, there was significant evidence of achieving the educational outcomes in Figure 3. Specifically:

1) Prior to 2005, no student achieved *any* of the outcomes at the *analysis*, *synthesis*, or *evaluation* levels. During this time, the only descriptions of the access-control logic available to students were research papers ([4], [5], [6], [7]) and our presentations. As mentioned in Section II, several versions of the access-control logic appear in those papers, and the undergraduates found them virtually impenetrable. These results in part spurred our decision to incorporate active-learning exercises into the 2005 version of the course.
2) In 2005, students were able to use the access-control logic for application outcomes **Ap1** and **Ap2**, but not **Ap3**; do analysis as described by outcomes **An1–An7**; do simple synthesis exercises related to **S1** and **S2**, but not **S3** and **S4** (the Saltzer and Schroeder [12] applications); and not able to do **E1**. In addition to the in-class exercises, we gave the class a draft textbook chapter that contained a single version of an access-control logic with provably sound inference rules. This chapter was written in a tutorial form with examples and exercises.
3) In 2006, students achieved all the educational outcomes listed in Figure 3 (where outcomes **Ap3** and **S3** on hardware descriptors were replaced by outcomes on delegation and proxies). Their achievement of evaluation outcome **E1** was still weak, but an improvement over previous years: the student proofs were reasonable, although some did contain errors. To this class, we had provided draft textbook chapters on the access-control logic, on basic access-control concepts, and on delegation.

In retrospect, we believe that having a textbook with examples and exercises was a significant help for students, in addition to the active-learning exercises introduced in lecture. We also recognize that instructors who might wish to follow our example could do so more easily with a textbook. In 2005, it was unclear to us that our approach would be sufficiently viable to justify writing a textbook. Based on our experiences, we are convinced that such a textbook is both possible and beneficial, and we are currently developing one [13].

Comprehension

C8: Translate and interpret descriptions using access control matrices into formulas in the access-control logic and vice versa.

Analysis

An8: When given an uncertified or a certified delegation, a request, and an access policy, you should be able to derive whether or not the request should be honored.

An9: When given a set of RBAC or SARBAC definitions, you should be able to determine the consequences of those definitions, such as a given role's administrative scope, administrative and non-administrative permissions, and whether a specific operation should be allowed.

Synthesis

S5: When given an access-control scenario, define a set of roles, role hierarchy, user and permission assignments, separation-of-duty relations, and SARBAC constraints to accurately reflect the scenario.

Evaluation

E2: When given a set of RBAC and SARBAC definitions, you should be able to judge whether they are consistent and (if not) identify all inconsistencies.

Fig. 15: Additional Educational Outcomes

VIII. Conclusion and Final Remarks

One question we have not yet addressed is: "Is a one-day 8-hour intensive short course the best way to teach access control rigorously?" Our answer is that it depends upon how much time one has to devote to the study of access control.

Our belief is that access control is central to any study of security and that as engineers and computer scientists we need to provide a mathematically rigorous treatment of the subject. Our experience leads us to believe that a one-day 8-hour intensive short course is about the *minimum* amount of time one can spend to deliver the educational outcomes listed in Figure 3. In particular, the kinds of analytical outcomes pertaining to rigorous descriptions of policies and derivations of access-control decisions really do require about four hours of syntax and semantics and four hours of access-control concepts. This effort amounts to approximately 2.5 weeks of class time in a standard-format class, not including homework.

If all one has is eight hours of class time, then what we have done as an intensive short course is effective. Nevertheless, it is far from complete.

The primary content difference between our 2005 and 2006 versions of the short course is in the day's final session: in 2005 we attempted to teach descriptor-based process isolation and sharing, whereas in 2006 we focused instead on delegation. The study of process isolation and sharing is essential to understanding computer security. Establishing the validity of delegations and policies pertaining to delegates is critical to any study of distributed or decentralized access control. Given only eight hours, one can have time for process isolation and sharing or for delegation, but not both.

Additional access-control topics would include role-based access control (RBAC) [14] and scoped administration of role-based access control (SARBAC) [15]. Reasoning about RBAC requires the addition of roles, partial orderings among principals, and notions of static and dynamic separation of duty. Extensions to the access-control logic to account for

these concepts are described in [11]. Finally, a more thorough treatment of access control would relate the results in the access-control calculus to classical models of access control such as HRU [16].

Figure 15 lists the additional educational outcomes for these additional topics. Our current graduate course, *CIS/CSE 774: Principles of Distributed Access Control*, is largely organized around the outcomes in Figures 3 and 15. Based on our experience with the ACE short course and CIS/CSE 774, we conclude that not only is a rigorous approach to teaching access control possible, but also necessary to address the increasing need for engineers and scientists to specify, design, build, and verify secure systems.

REFERENCES

[1] Dan Carnevale, "Basic training for anti-hackers: An intensive summer program drills students on cybersecurity skills," *The Chronicle of Higher Education*, September 23 2005.

[2] Kamal Jabbour and Susan Older, "The advanced course in engineering on cyber security: A learning community for developing cyber-security leaders," in *Proceedings of the Sixth Workshop on Education in Computer Security*, July 2004.

[3] Butler Lampson, "Computer security in the real world," *IEEE Computer*, vol. 37, no. 6, pp. 37–46, June 2004.

[4] Butler Lampson, Martin Abadi, Michael Burrows, and Edward Wobber, "Authentication in distributed systems: Theory and practice," *ACM Transactions on Computer Systems*, vol. 10, no. 4, pp. 265–310, November 1992.

[5] John Howell and David Kotz, "A formal semantics for SPKI," Tech. Rep. TR2000-363, Department of Computer Science, Dartmouth College, Hanover, NH 03755-3510, March 2000.

[6] Martin Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin, "A calculus for access control in distributed systems," *ACM Transactions on Programming Languages and Systems*, vol. 15, no. 4, pp. 706–734, September 1993.

[7] Edward Wobber, Martin Abadi, Michael Burrows, and Butler Lampson, "Authentication in the TAOS operating system," *ACM Transactions on Computer Systems*, vol. 12, no. 1, pp. 3–32, February 1994.

[8] Robert M. Diamond, *Designing & Assessing Courses & Curricula: A Practical Guide*, Jossey-Boss, revised edition, 1998.

[9] Benjamin S. Bloom, *The Taxonomy of Educational Objectives: Affective and Cognitive Domains*, David McKay, New York, 1974.

[10] Thumrongsak Kosiyatrakul, Susan Older, Polar Humenn, and Shiu-Kai Chin, "Implementing a calculus for distributed access control in higher order logic and HOL," in *Second International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security*, Vladimir Gorodetsky, Leonard Popyack, and Victor Skormin, Eds., 2003, vol. 2776.

[11] Thumrongsak Kosiyatrakul, Susan Older, and Shiu-Kai Chin, "A modal logic for role-based access control," in *Third International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security*, Vladimir Gorodetsky, Igor Kotenko, and Victor Skormin, Eds. 2005, number 3685 in Lecture Notes in Computer Science, pp. 179–193, Springer.

[12] Jerome Saltzer and Michael Schroeder, "The protection of information in computer systems," *Proceedings of IEEE*, 1975.

[13] Shiu-Kai Chin and Susan Older, *A Mathematical Introduction to Access Control*, CRC Press, to appear in 2008.

[14] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli, "Proposed nist standard for role-based access control," *ACM Transaction on Information and System Security*, 2001.

[15] J. Cramton and G. Loizou, "Administrative scope: A foundation for role-based administrative models," *ACM Transactions on Information and System Security*, vol. 6, no. 2, pp. 201–231, 2003.

[16] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman, "Protection in operating systems," *Communications of the ACM*, vol. 19, no. 8, pp. 461–470, August 1976.