

Chapter 2

A Language for Access Control

In this chapter we introduce the core logic that relates *principals* to the statements they make. The term *principal* refers to any person, process, or agent that attempts to exercise an action on an object (i.e., assert its *rights*). The statements we care about include access requests (e.g., Alice states her desire to read a certain file), as well as statements of rights (e.g., the Department of Motor Vehicles states that Bob is authorized to drive a truck), authority, and jurisdiction (e.g., the Department of Motor Vehicles has the jurisdiction to authorize Bob to drive a truck), and even the association of keys with principals (e.g., a specific 128-bit encryption key is associated with Carol).

In subsequent chapters, we will express and reason about a variety of access-control concepts using this logic. The purpose of this chapter and the next is to provide a primer to the logic itself. There are three important components of any logic: the *syntax* (“what do the formulas look like?”), the *semantics* (“what do the formulas mean?”), and the *logical rules* (“how can I manipulate formulas to carry out my reasoning?”). We introduce the first two of these components in this chapter, leaving the inference rules for Chapter 3. However, we begin by reviewing important concepts and notation from discrete mathematics that we will use throughout this book, including sets, relations, and certain operations on them.

2.1 Sets and Relations

We assume a working knowledge of discrete mathematics and propositional logic, and thus we do not provide a detailed introduction to them. Instead, we briefly introduce the notation that we will be using, and we review some simple and common approaches for proving properties about sets and relations.

The exercises in this section can be used as a guide to determine whether you have the requisite knowledge of discrete mathematics and propositional logic. Also, the references at the end of this chapter contain several suggested texts to consult if you need a detailed introduction.

2.1.1 Notation

We use standard set notation. Set elements are contained within braces, as in $\{e_0, e_1, \dots, e_n\}$. The *empty set* (i.e., the set that has no elements) is denoted as either $\{\}$ or \emptyset . Set *union* is denoted by \cup , set *intersection* is denoted by \cap , and set *difference* is denoted by $-$, as illustrated by the following simple examples:

$$\begin{aligned}\{1, 2\} \cup \{1, 3, 4\} &= \{1, 2, 3, 4\} \\ \{1, 2, 3\} \cap \{2, 3, 4, 5\} &= \{2, 3\} \\ \{2, 3, 4, 5\} - \{1, 4, 5\} &= \{2, 3\}.\end{aligned}$$

The order in which elements are listed in a set is unimportant. Thus, for example, $\{1, 2\}$ and $\{2, 1\}$ denote the same set.

We write $x \in S$ to indicate that x is an element of the set S ; likewise, $x \notin S$ indicates that x is not an element of the set S .

A set S is a *subset* of T , written $S \subseteq T$, if every element of S is also an element of T . The *power set* of a set S , written $\mathcal{P}(S)$, is the set of all subsets of S . For example, the power set of $\{\text{red}, \text{blue}\}$ is

$$\mathcal{P}(\{\text{red}, \text{blue}\}) = \{\{\}, \{\text{red}\}, \{\text{blue}\}, \{\text{red}, \text{blue}\}\}.$$

Note that the empty set is a subset of every set (i.e., $\emptyset \subseteq S$ for all S), and every set S is a subset of itself (i.e., $S \subseteq S$). Thus, \emptyset and S are always elements of $\mathcal{P}(S)$.

The *Cartesian product* of sets A and B (written $A \times B$) is the set of ordered pairs whose first component is drawn from A and whose second component is drawn from B :

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}.$$

A *binary relation* is simply a set $R \subseteq A \times B$ of pairs whose first components are drawn from A and whose second components are drawn from B . We say that R is a *binary relation over (or on) A* when $R \subseteq A \times A$.

The *identity relation* on a set A is the relation $\text{id}_A \subseteq A \times A$ defined by:

$$\text{id}_A = \{(a, a) \mid a \in A\}.$$

The *composition* of relations $R_1 \subseteq A \times B$ and $R_2 \subseteq B \times C$ is the relation $R_1 \circ R_2 \subseteq A \times C$ defined as follows:

$$R_1 \circ R_2 = \{(x, z) \mid \text{there exists } y \text{ such that } ((x, y) \in R_1 \text{ and } (y, z) \in R_2)\}.$$


Finally, given a relation $R \subseteq A \times B$ and an element $a \in A$, we define $R(a)$ to be the *image of R under a*:

$$R(a) = \{b \in B \mid (a, b) \in R\}.$$

That is, $R(a)$ is the set of elements in B related to a by the relation R . For example, if S is the relation

$$S = \{(1, 2), (2, 3), (2, 4), (3, 5), (3, 1), (4, 1), (5, 2)\},$$

then $S(2) = \{3, 4\}$ and $S(3) = \{5, 1\}$.

 **Exercise 2.1.1** Let T and U be relations over the set $A = \{1, 2, 3, 4\}$, as follows:

$$T = \{(1, 1), (2, 1), (3, 3), (4, 4), (3, 4)\}$$

$$U = \{(2, 4), (1, 3), (3, 3), (3, 2)\}.$$

Calculate the following sets:

a. $\mathcal{P}(\{x, y, z\})$

b. $U(3)$

c. $U(4)$

d. $T \cup U$

e. $T \cap U$

f. $T - U$

g. $U \circ T$

h. $T \circ U$

2.1.2 Approaches for Mathematical Proofs

Throughout this book, our examples and exercises include mathematical proofs involving sets, relations, and relationships among them. We highlight some common approaches for structuring these proofs.

Recall that a set S is a subset of T if every element of S is also an element of T . Thus, to prove that some set A is a subset of B , it suffices to demonstrate that any arbitrary element of A is necessarily also an element of B . The following simple example demonstrates this proof approach.

Example 2.1

Property: The subset relation is *transitive*: that is, if $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$.

Proof: Consider sets A, B, C such that $A \subseteq B$ and $B \subseteq C$, and consider an arbitrary element $x \in A$; we need to show that $x \in C$ as well.

Because $A \subseteq B$, we know that $x \in B$; because $B \subseteq C$, it follows that $x \in C$ as necessary. \diamond

Two sets A and B are equal (i.e., contain exactly the same elements) provided that each is a subset of the other. This fact forms the foundation for another standard proof approach: to prove that $A = B$, it suffices to demonstrate that $A \subseteq B$ and that $B \subseteq A$. The following example, which includes a useful property for subsequent sections, demonstrates this proof approach.

Example 2.2

Property: Suppose that R and T are arbitrary relations over a set A and that Y is a set; then the following equality holds:

$$\{x \mid (R \cup T)(x) \subseteq Y\} = \{x \mid R(x) \subseteq Y\} \cap \{x \mid T(x) \subseteq Y\}.$$

Proof: Consider arbitrary relations R, T and an arbitrary set Y . Our analysis involves two steps:

1. Consider an arbitrary element $a \in \{x \mid (R \cup T)(x) \subseteq Y\}$: thus $(R \cup T)(a) \subseteq Y$. By definition,

$$\begin{aligned} (R \cup T)(a) &= \{b \mid (a, b) \in R \cup T\} \\ &= \{b \mid (a, b) \in R\} \cup \{b \mid (a, b) \in T\} \\ &= R(a) \cup T(a). \end{aligned}$$

Therefore, $R(a) \cup T(a) \subseteq Y$, and hence we also have that $R(a) \subseteq Y$ and $T(a) \subseteq Y$. It follows that $a \in \{x \mid R(x) \subseteq Y\}$ and $a \in \{x \mid T(x) \subseteq Y\}$, and therefore $a \in \{x \mid R(x) \subseteq Y\} \cap \{x \mid T(x) \subseteq Y\}$. Because a was an arbitrary element of $\{x \mid (R \cup T)(x) \subseteq Y\}$, we have shown that

$$\{x \mid (R \cup T)(x) \subseteq Y\} \subseteq \{x \mid R(x) \subseteq Y\} \cap \{x \mid T(x) \subseteq Y\}.$$

2. Consider an arbitrary element $a \in \{x \mid R(x) \subseteq Y\} \cap \{x \mid T(x) \subseteq Y\}$. It follows that $R(a) \subseteq Y$ and $T(a) \subseteq Y$, and hence $R(a) \cup T(a) \subseteq Y$. Because $R(a) \cup T(a) = (R \cup T)(a)$, we also have that $(R \cup T)(a) \subseteq Y$, and thus $a \in \{x \mid (R \cup T)(x) \subseteq Y\}$. Because a was an arbitrary element of $\{x \mid R(x) \subseteq Y\} \cap \{x \mid T(x) \subseteq Y\}$, we have shown that

$$\{x \mid R(x) \subseteq Y\} \cap \{x \mid T(x) \subseteq Y\} \subseteq \{x \mid (R \cup T)(x) \subseteq Y\}.$$

Having demonstrated that each set is a subset of the other, we have shown that

$$\{x \mid (R \cup T)(x) \subseteq Y\} = \{x \mid R(x) \subseteq Y\} \cap \{x \mid T(x) \subseteq Y\}. \quad \diamond$$

One often encounters statements that indicate a given property (say, *Property 1*) is true *if and only if* another property (say, *Property 2*) is true. Proving that an “if and only if” statement is true involves two steps: (1) demonstrating that, whenever *Property 1* holds, *Property 2* must also hold; and (2) demonstrating that, whenever *Property 2* holds, *Property 1* must also hold. The following example demonstrates this proof approach to prove a property that will be useful for calculations in subsequent sections.

Example 2.3

Property: Let A, X , and Y be sets such that X and Y are both subsets of A . Then $A \subseteq (A - X) \cup Y$ if and only if $X \subseteq Y$.

Proof: Our analysis involves two steps:

1. The “forward” direction: suppose $A \subseteq (A - X) \cup Y$, and consider any $x \in X$. Since $X \subseteq A$, $x \in A$, and thus $x \notin A - X$. Therefore, x must be an element of Y . Since x was arbitrary, $X \subseteq Y$.
2. The “reverse” direction: suppose $X \subseteq Y$, and consider any $a \in A$. If $a \in X$, then by the definition of subset, $a \in Y$, and hence $a \in (A - X) \cup Y$ as necessary; if, instead, $a \notin X$, then $a \in (A - X)$ and therefore $a \in (A - X) \cup Y$. Since a was arbitrary, $A \subseteq (A - X) \cup Y$.

Having shown that each property implies the other, we have demonstrated that $A \subseteq (A - X) \cup Y$ if and only if $X \subseteq Y$. \diamond

The exercises that follow include properties that will be useful for exercises in subsequent sections.



Exercise 2.1.2 Prove that, for all relations R, S, T , the following property holds:

$$\text{If } R \subseteq S, \text{ then } R \circ T \subseteq S \circ T.$$



Exercise 2.1.3 Prove that, for all relations R, S, T , the following property holds:

$$(R \cup S) \circ T = (R \circ T) \cup (S \circ T).$$



Exercise 2.1.4 Prove that, for all sets A, B, C , the following property holds:

$$((A - B) \cup C) \cap ((A - C) \cup B) = (B \cap C) \cup ((A - B) \cap (A - C)).$$



Exercise 2.1.5 Prove that, for all relations R, S , sets X , and items u , the following property holds:

$$(R \circ S)(u) \subseteq X \text{ if and only if } R(u) \subseteq \{y \mid S(y) \subseteq X\}.$$

2.2 Syntax

Our goal is to define a logic for expressing access-control policies and analyzing their ramifications. However, to do so, we must first identify the most primitive concepts that we wish to express.

The first step is to introduce the *syntax* of the logic: that is, we must specify what the formulas of the logic look like. To do so, we make use of *BNF (Backus-Naur Form) specifications*, which provide a way to state syntactic rules precisely

and unambiguously. We explain how BNF specifications work through the following example, which describes the syntax for a simple language of arithmetic expressions:

$$\begin{aligned}
 \mathbf{AExp} &::= \mathbf{BinNumber} \\
 \mathbf{AExp} &::= (\mathbf{AExp} + \mathbf{AExp}) \\
 \mathbf{AExp} &::= (\mathbf{AExp} * \mathbf{AExp}) \\
 \mathbf{BinNumber} &::= \mathbf{Bit} \\
 \mathbf{BinNumber} &::= \mathbf{Bit} \mathbf{BinNumber} \\
 \mathbf{Bit} &::= 0 \\
 \mathbf{Bit} &::= 1
 \end{aligned}$$

The items in boldface (e.g., **AExp** and **BinNumber**) are called *non-terminal symbols* and represent *syntactic categories* (i.e., collections of syntactic expressions). The symbols “(”, “)”, “+”, “*”, “0”, and “1” are called *terminal symbols* (or *terminals*) and correspond to actual symbols in the syntax. The notation “::= ” is meta-notation for specifying *production rules* for the syntax. In general, each line of form

$$\mathbf{AExp} ::= \textit{right-hand-side}$$

provides a rule for creating new elements of **AExp** (and likewise for **BinNumber** and **Bit**). A *syntactic derivation* is a sequence of rewriting steps: each step uses one of the production rules to replace one occurrence of a nonterminal symbol. The syntactic derivation is complete when there are no more nonterminal symbols to replace.

The following two examples contain syntactic derivations that respectively demonstrate that 101 belongs to the syntactic category **BinNumber** and that $(11 * (1 + 10))$ belongs to the syntactic category **AExp**. Note that a given piece of syntax (such as 101) may have multiple derivations possible, corresponding to choosing different nonterminals to expand at a given step. The order in which nonterminal symbols are replaced is not important; what *is* important is that at least one syntactic derivation must exist for a given piece of syntax in order for it to be considered *well formed*.

Example 2.4

The following derivation demonstrates that 101 belongs to the syntactic category **BinNumber**:

$$\begin{aligned}
 \mathbf{BinNumber} &\rightsquigarrow \mathbf{Bit} \mathbf{BinNumber} \\
 &\rightsquigarrow 1 \mathbf{BinNumber} \\
 &\rightsquigarrow 1 \mathbf{Bit} \mathbf{BinNumber} \\
 &\rightsquigarrow 1 \mathbf{Bit} \mathbf{Bit} \\
 &\rightsquigarrow 10 \mathbf{Bit} \\
 &\rightsquigarrow 101
 \end{aligned}$$

◇

Example 2.5

The following derivation demonstrates that $(11 * (1 + 10))$ belongs to the syntactic category **AExp**:

$$\begin{aligned}
\mathbf{AExp} &\rightsquigarrow (\mathbf{AExp} * \mathbf{AExp}) \\
&\rightsquigarrow (\mathbf{AExp} * (\mathbf{AExp} + \mathbf{AExp})) \\
&\rightsquigarrow (\mathbf{AExp} * (\mathbf{BinNumber} + \mathbf{AExp})) \\
&\rightsquigarrow (\mathbf{AExp} * (\mathbf{Bit} + \mathbf{AExp})) \\
&\rightsquigarrow (\mathbf{AExp} * (1 + \mathbf{AExp})) \\
&\rightsquigarrow (\mathbf{AExp} * (1 + \mathbf{BinNumber})) \\
&\rightsquigarrow (\mathbf{AExp} * (1 + \mathbf{Bit} \mathbf{BinNumber})) \\
&\rightsquigarrow (\mathbf{AExp} * (1 + 1 \mathbf{Bit})) \\
&\rightsquigarrow (\mathbf{AExp} * (1 + 10)) \\
&\rightsquigarrow (\mathbf{BinNumber} * (1 + 10)) \\
&\rightsquigarrow (\mathbf{Bit} \mathbf{BinNumber} * (1 + 10)) \\
&\rightsquigarrow (\mathbf{Bit} \mathbf{Bit} * (1 + 10)) \\
&\rightsquigarrow (1 \mathbf{Bit} * (1 + 10)) \\
&\rightsquigarrow (11 * (1 + 10)) \quad \diamond
\end{aligned}$$

Multiple production rules for the same syntactic category can be combined into a single rule by using the meta-symbol “/”, which separates possible alternatives. For example, the three production rules for the **AExp** syntactic category could instead be specified as follows:

$$\mathbf{AExp} ::= \mathbf{BinNumber} / (\mathbf{AExp} + \mathbf{AExp}) / (\mathbf{AExp} * \mathbf{AExp})$$

This production rule states that an element of the syntactic category **AExp** has either the form **BinNumber**, the form $(\mathbf{AExp} + \mathbf{AExp})$, or the form $(\mathbf{AExp} * \mathbf{AExp})$.

2.2.1 Principal Expressions

Principals are the major actors in a system. They are the entities that make requests, and the class of principals includes (but is not limited to) people, processes, cryptographic keys, personal identification numbers (PINs), userid–password pairs, and so on.

In general, principals may be either *simple* or *compound*. A simple principal is an entity that cannot be decomposed further: individuals, cryptographic keys, and userid–password pairs are all simple principals. We define **PName** to be the collection of all simple principal names, which can be used to refer to any simple principal. For example, the following are all allowable principal names: *Alice*, *Bob*, the key K_{Alice} , the PIN 1234, and the userid–password pair $\langle a_{lice}, bAdPsWd! \rangle$.

Compound principals are abstract entities that connote a combination of principals: for example, “the President in conjunction with Congress” connotes an abstract

principal comprising both the President and Congress. Intuitively, such a principal makes exactly those statements that are made by *both* the President and Congress. Similarly, “the reporter quoting her source” connotes an abstract principal that comprises both the reporter and her source. Intuitively, a statement made by such a principal represents a statement that the reporter is (rightly or wrongly) attributing to his source.

The set **Princ** of *all* principal expressions is given by the following BNF specification:

$$\mathbf{Princ} ::= \mathbf{PName} / \mathbf{Princ} \ \& \ \mathbf{Princ} / \mathbf{Princ} \ | \ \mathbf{Princ}$$

That is, a principal expression is either a simple name, an expression of form $P \ \& \ Q$ (where P and Q are both principal expressions), or an expression of form $P \ | \ Q$ (where, again, P and Q are both principal expressions).

The principal expression $P \ \& \ Q$ denotes the abstract principal “ P in conjunction with Q ,” while $P \ | \ Q$ denotes the abstract principal “ P quoting Q .” Thus, the expression *President & Congress* denotes the abstract principal “the President together with Congress,” and *Reporter | Source* denotes the abstract principal “the reporter quoting her source.” In subsequent chapters, we will introduce a few more compound principal expressions to cover other important concepts, such as *delegation*. However, we shall see that those additional expressions are merely special cases of quoting and conjunctive principals.

Parentheses can be added to disambiguate compound principal expressions. For example, $(Sal \ \& \ Ted) \ | \ Uly$ denotes the conjunctive principal *Sal & Ted* quoting the principal *Uly*. In contrast, $Sal \ \& \ (Ted \ | \ Uly)$ denotes the principal *Sal* acting in concert with the compound principal *Ted | Uly*. The standard convention in such expressions is that $\&$ binds more tightly than $|$, so that (for example) $Sal \ \& \ Ted \ | \ Uly$ is equivalent to $(Sal \ \& \ Ted) \ | \ Uly$. However, in this book we shall always include the parentheses necessary to disambiguate principal expressions. Both $\&$ and $|$ are associative operators, and hence it is unnecessary to include parentheses around principal expressions such as *Fritz & Hans & Leon* or *Terry | Kitty | Sandy*.

2.2.2 Access-Control Statements

Ultimately, we are concerned with being able to determine with precision and accuracy which access requests from which principals should be granted, and which should be denied. Thus, we need to be able to express our assumptions and our expectations, such as which authorities we trust, which principals should be granted access to which objects, and so on. Toward this end, we introduce a language of statements that will allow us to express these sorts of concepts.

The simplest statements are basic requests, such as “read file *foo*” or “modify file *bar*.” We will represent such statements in our logic by *propositional variables*, in much the same way that statements such as “it is raining” can be represented in

propositional logic.¹

However, a simple request such as “read file *foo*” by itself is generally insufficient for determining whether or not to grant the request. At a minimum, we need some way of accounting for the *source* of the request. In our logic, we can associate requests (and other statements) with their source by statements of the form

$$P \text{ says } \phi,$$

where P is a principal and ϕ is a specific statement. For example, if rff is a propositional variable representing the request “read file *foo*,” then we can represent Deena’s request to read the file *foo* by the statement

$$\text{Deena says } rff.$$

The *says* operator can also ascribe non-request statements to particular principals. For example, we may wish to express that Rob believes (or has stated) that Deena is making a request to read file *foo*. We can express such a concept in the logic by the statement

$$\text{Rob says } (\text{Deena says } rff).$$

Access policies specify which principals are authorized to access particular objects. Such authorizations can be expressed in our logic by statements of the form

$$P \text{ controls } \phi,$$

where P is a principal and ϕ is a specific statement. Thus, for example, we can express Deena’s entitlement to read the file *foo* as the statement

$$\text{Deena controls } rff.$$

Note that this statement is different from Deena’s request to read the file. This statement merely says that *if* Deena says that it’s a good idea to read file *foo* (i.e., if Deena makes the request), then it is indeed a good idea to read file *foo* (i.e., the request should be granted).

In any given situation, there are particular authorities who are deemed to have *jurisdiction* over particular statements: that is, we will unquestioningly believe certain statements that they make. For example, Anna’s New York State driver’s license in effect represents a statement from the New York State Department of Motor Vehicles (DMV) that vouches for Anna’s ability to drive a car. Anyone who accepts the DMV’s jurisdiction for making that statement will, upon seeing Anna’s valid driving license, accept as truth that she is entitled to drive. Like authorizations, jurisdiction is represented in our logic by statements of the form

$$P \text{ controls } \phi,$$

¹Technically, propositions are statements that are interpreted to be either true or false, such as the proposition “it is raining.” If we view an access request such as “read file *foo*” as shorthand for “it would be a good thing to let me read file *foo*,” then we indeed have a legitimate proposition.

where P is a principal with jurisdiction over the statement ϕ .

Finally, we would like a way to be able to make statements about the relative trustedness of different principals, which we do through the use of the operator \Rightarrow (pronounced “speaks for”). Intuitively, the statement

$$P \Rightarrow Q$$

describes a proxy relationship between the two principals P and Q such that any statement made by P can also be safely attributed to Q (i.e., acted upon as if Q had made it).

In addition to the sorts of statements mentioned above, we make use of the standard logical operators: negation ($\neg\phi$), conjunction ($\phi_1 \wedge \phi_2$), disjunction ($\phi_1 \vee \phi_2$), implication ($\phi_1 \supset \phi_2$), and equivalence ($\phi_1 \equiv \phi_2$). Having provided an informal overview of our logic, we give a formal definition of our logic’s syntax in the next subsection.

2.2.3 Well-Formed Formulas

We give the name **PropVar** to the collection of all propositional variables, and we typically use lowercase identifiers (such as p, q, r) to range over this set. We can construct more interesting statements from these simple propositional variables and from our set **Princ** of principal expressions. The set **Form** of all well-formed expressions (also known as *well-formed formulas*) in our language is given by the following BNF specification:

$$\begin{aligned} \mathbf{Form} ::= & \mathbf{PropVar} / \neg \mathbf{Form} / (\mathbf{Form} \vee \mathbf{Form}) / \\ & (\mathbf{Form} \wedge \mathbf{Form}) / (\mathbf{Form} \supset \mathbf{Form}) / (\mathbf{Form} \equiv \mathbf{Form}) / \\ & (\mathbf{Princ} \Rightarrow \mathbf{Princ}) / (\mathbf{Princ} \text{ says } \mathbf{Form}) / (\mathbf{Princ} \text{ controls } \mathbf{Form}) \end{aligned}$$

This BNF specification provides all of the information necessary to determine the structure of well-formed formulas in our access-control logic. The following are examples of well-formed formulas:

$$\begin{aligned} & r \\ & ((\neg q \wedge r) \supset s) \\ & (Jill \text{ says } (r \supset (p \vee q))) \end{aligned}$$

Convention: Throughout this book, we will distinguish between principal names and propositional variables through capitalization. Specifically, we will use *capitalized* identifiers—such as *Josh* and *Reader*—for simple principal names. We will use *lowercase* identifiers—such as r , *write*, and *rff*—for propositional variables.

Example 2.6

The following syntactic derivation demonstrates that $(Jill\ says\ (r \supset (p \vee q)))$ is a well-formed formula:

$$\begin{aligned}
\mathbf{Form} &\rightsquigarrow (\mathbf{Princ\ says\ Form}) \\
&\rightsquigarrow (\mathbf{PName\ says\ Form}) \\
&\rightsquigarrow (Jill\ \mathbf{says\ Form}) \\
&\rightsquigarrow (Jill\ \mathbf{says\ (Form \supset Form)}) \\
&\rightsquigarrow (Jill\ \mathbf{says\ (PropVar \supset Form)}) \\
&\rightsquigarrow (Jill\ \mathbf{says\ (r \supset Form)}) \\
&\rightsquigarrow (Jill\ \mathbf{says\ (r \supset (Form \vee Form))}) \\
&\rightsquigarrow (Jill\ \mathbf{says\ (r \supset (PropVar \vee Form))}) \\
&\rightsquigarrow (Jill\ \mathbf{says\ (r \supset (p \vee Form))}) \\
&\rightsquigarrow (Jill\ \mathbf{says\ (r \supset (p \vee PropVar))}) \\
&\rightsquigarrow (Jill\ \mathbf{says\ (r \supset (p \vee q))}) \quad \diamond
\end{aligned}$$

In contrast, the following examples are *not* well-formed formulas, for the reasons stated:


- *Orly & Mitch* is a principal expression, but not an access-control formula.
- $\neg Orly$, because *Orly* is a principal expression, not an access-control formula; the negation operator \neg must precede an access-control formula.
- $(Orly \Rightarrow (p \wedge q))$, because $(p \wedge q)$ is not a principal expression: the speaks-for operator \Rightarrow must appear between two principal expressions.
- $(Orly\ controls\ Mitch)$, because *Mitch* is a principal expression, not an access-control formula; the controls operator requires its second argument to be an access-control formula.

The parentheses ensure that the grammar is completely unambiguous, but their excessive proliferation can make the language cumbersome to use. Thus, we typically omit the outermost parentheses, and we occasionally also omit additional parentheses according to standard conventions for operator precedence: \neg binds most tightly, followed in order by \wedge , \vee , \supset , and \equiv . Thus, for example, the formula $p \supset q \wedge r$ is an abbreviation of $(p \supset (q \wedge r))$. Likewise, the formulas $((\neg q \wedge r) \supset s)$ and $(Jill\ says\ (r \supset (p \vee q)))$ can be abbreviated respectively as follows:


$$\neg q \wedge r \supset s, \quad Jill\ says\ (r \supset p \vee q).$$

When the same binary operator appears multiple times in a formula—for example, $p \supset q \supset r$ —the parentheses associate from left to right: $(p \supset q) \supset r$. The operators *says* and *controls* bind even more tightly than \wedge , and thus have as small a scope as

possible. For example, *Kent* says $r \vee p \supset q$ is equivalent to $((\text{Kent says } r) \vee p) \supset q$, and similarly for controls. Because \Rightarrow is an operator that relates only principal expressions (as opposed to logical formulas), its use is always unambiguous.

 **Exercise 2.2.1** Which of the following are well-formed formulas in the access-control logic? Support your answers by appealing to the BNF specification.

- a. $((p \wedge \neg q) \supset (\text{Cal controls } r))$
- b. $((\text{Gin} \Rightarrow r) \wedge q)$
- c. $(\text{Mel} \mid \text{Ned says } (r \supset t))$
- d. $(\neg t \Rightarrow \text{Sal})$
- e. $(\text{Ulf controls } (\text{Vic} \mid \text{Wes} \Rightarrow \text{Tor}))$
- f. $(\text{Pat controls } (\text{Quint controls } (\text{Ryne says } s)))$

 **Exercise 2.2.2** Fully parenthesize each of the following formulas:

- a. $p \supset \neg q \vee r \supset s$
- b. $\neg p \supset r \equiv q \vee r \supset t$
- c. $X \text{ controls } t \vee s \supset Y \text{ says } q \supset r$
- d. $Cy \text{ says } q \wedge Di \text{ controls } p \supset r$
- e. $Ike \Rightarrow Jan \wedge Kai \ \& \ Lee \text{ controls } q \wedge r$

2.3 Semantics

Although we provided an informal reading of the logical formulas in the previous section, we have not yet provided sufficient details to enable precise or rigorous use of the logic. In the next chapter, we introduce logical rules that we can use to reason about a variety of access-control situations. However, as with any logic, several important questions arise:

What statements are true in this logic, and how do we know? What does a given statement really mean? How do we know that the inference rules are trustworthy? Under what conditions can we add new inference rules and guarantee that the logic's trustworthiness is preserved?

| p | q | r | $p \wedge q$ | $(p \wedge q) \supset r$ |
|-------|-------|-------|--------------|--------------------------|
| true | true | true | true | true |
| true | true | false | true | false |
| true | false | true | false | true |
| true | false | false | false | true |
| false | true | true | false | true |
| false | true | false | false | true |
| false | false | true | false | true |
| false | false | false | false | true |

Table 2.1: Truth table for the propositional formula $(p \wedge q) \supset r$

The key to answering these questions for any logic is to have rigorous, mathematical semantics that define precisely what a given statement means. These formal semantics provide a basis by which one can independently assess the trustworthiness of a logical system. For example, in propositional logic, the formal meaning of a statement such as

$$(p \wedge q) \supset r$$

can be calculated using a truth table, as illustrated in Table 2.1. Each line in the truth table corresponds to a particular interpretation of the propositional variables (i.e., a mapping of variables to specific truth values). Truth tables calculate the meaning of larger formulas in a *syntax-directed* way, based on the meanings of their components: for example, the meaning of $(p \wedge q) \supset r$ for a given interpretation is calculated using the meanings of $p \wedge q$ and r , as well as a specific rule for the operator \supset . A propositional-logic formula is a *tautology*—and therefore safe to use as an axiom of the system—if it is true for *every* possible interpretation of the propositional variables.

These same core ideas apply to the semantics for our access-control logic. Because we must account for the interpretation of *principals* in addition to *propositional variables*, the semantics requires a little more structure than truth tables provide. We can find this additional structure in the form of *Kripke structures*.

2.3.1 Kripke Structures

Kripke structures are useful models for analyzing a variety of situations. They are commonly used to provide semantics for modal and temporal logics, providing a basis for automated model checking.

Definition 2.1 A Kripke structure \mathcal{M} is a three-tuple $\langle W, I, J \rangle$, where:

- W is a nonempty set, whose elements are called worlds.
- $I : \mathbf{PropVar} \rightarrow \mathcal{P}(W)$ is an interpretation function that maps each propositional variable to a set of worlds.

- $J : PName \rightarrow \mathcal{P}(W \times W)$ is a function that maps each principal name to a relation on worlds (i.e., a subset of $W \times W$). ■

Before we look at some examples of Kripke structures, a few comments about this definition are in order. First, the concept of *worlds* is an abstract one. In reality, W is simply a set: its contents (whatever they may be) are called worlds. In many situations, the notion of worlds corresponds to the notion of system states or to the concept of possible alternatives.

Second, the functions I and J provide meanings (or *interpretations*) for our propositional variables and simple principals. These meanings will form the basis for our semantics of arbitrary formulas in our logic. Intuitively, $I(p)$ is the set of worlds in which we consider p to be true. $J(A)$ is a relation that describes how the simple principal A views the relationships between worlds: each pair $(w, w') \in J(A)$ indicates that, when the current world is w , principal A believes it possible that the current world is w' .

For illustration purposes, we introduce some examples of Kripke structures. The first example provides some intuition as to what the interpretation functions I and J represent, illustrating how each relation $J(P)$ might reflect principal P 's understanding of the universe.

Example 2.7

Consider the situation of three young children (*Flo*, *Gil*, and *Hal*), who are being looked after by an overprotective babysitter. This babysitter will let them go outside to play only if the weather is both sunny and sufficiently warm.

To keep things simple, let us imagine that there are only three possible situations: it is sunny and warm, it is sunny but cool, or it is not sunny. We can represent these possible alternatives with a set of three worlds: $W_0 = \{sw, sc, ns\}$.

We use the propositional variable g to represent the proposition “The children can go outside.” The baby sitter’s overprotectiveness can be represented by any interpretation function

$$I_0 : \mathbf{PropVar} \rightarrow \mathcal{P}(\{sw, sc, ns\})$$

for which $I_0(g) = \{sw\}$. That is, the proposition g (“the children can go outside”) is true only in the world sw (i.e., when the weather is both sunny and warm).

Now, the children themselves are standing by the window, trying to determine whether or not they’ll be allowed to go outside. *Gil*, who is tall enough to see the outdoor thermometer, possesses perfect knowledge of the situation, as he will be able to determine whether it is both sunny and sufficiently warm. This perfect knowledge corresponds to a possible-worlds relation

$$J_0(Gil) = \{(sw, sw), (sc, sc), (ns, ns)\}.$$

Whatever the current situation is, *Gil* has the correct understanding of the situation. (Note that $J_0(Gil)$ is the identity relation id_{W_0} over the set W_0 .)

In contrast, *Flo* is too short to see the outdoor thermometer, and thus she cannot distinguish between the “sunny and warm” and “sunny and cool” alternatives. This uncertainty corresponds to a possible-worlds relation

$$J_0(\textit{Flo}) = \{(sw, sw), (sw, sc), (sc, sw), (sc, sc), (ns, ns)\}.$$

Thus, for example, if the current situation is “sunny and warm” (i.e., *sw*), *Flo* considers both “sunny and warm” and “sunny and cool” as legitimate possibilities. That is, $J_0(\textit{Flo})(sw) = \{sw, sc\}$.

Finally, *Hal* is too young to understand that it can be simultaneously sunny and cool: he believes that the presence of the sun automatically makes it warm outside. His confusion corresponds to a possible-worlds relation

$$J_0(\textit{Hal}) = \{(sw, sw), (sc, sw), (ns, ns)\}.$$

Whenever the actual weather is sunny and cool, *Hal* believes it to be sunny and warm: $J_0(\textit{Hal})(sc) = \{sw\}$.

The tuple $\langle W_0, I_0, J_0 \rangle$ forms a Kripke structure. ◇

The next example introduces a Kripke structure that does not necessarily reflect any particular scenario or vignette. Rather, the Kripke structure is merely a three-tuple that contains a set and two functions that match the requirements of Definition 2.1.

Example 2.8

Let $W_1 = \{w_0, w_1, w_2\}$ be a set of worlds, and let $I_1 : \mathbf{PropVar} \rightarrow \mathcal{P}(W_1)$ be the interpretation function defined as follows²:

$$\begin{aligned} I_1(q) &= \{w_0, w_2\}, \\ I_1(r) &= \{w_1\}, \\ I_1(s) &= \{w_1, w_2\}. \end{aligned}$$

In addition, let $J_1 : \mathbf{PName} \rightarrow \mathcal{P}(W_1 \times W_1)$ be the function defined as follows³:

$$\begin{aligned} J_1(\textit{Alice}) &= \{(w_0, w_0), (w_1, w_1), (w_2, w_2)\}, \\ J_1(\textit{Bob}) &= \{(w_0, w_0), (w_0, w_1), (w_1, w_2), (w_2, w_1)\}. \end{aligned}$$

The three-tuple $\langle W_1, I_1, J_1 \rangle$ is a Kripke structure. Intuitively, proposition *q* is true in worlds w_0 and w_2 , *r* is true in world w_1 , and *s* is true in worlds w_1 and w_2 . All other propositions are false in all worlds. ◇

²In this example and those that follow, we adopt the convention of specifying only those propositional variables that the interpretation function maps to nonempty sets of worlds. Thus, for any propositional variable *p* not explicitly mentioned, we assume that $I_1(p) = \emptyset$.

³We adopt a similar convention for principal-mapping functions *J*: for any principal name *A* for which $J(A)$ is not explicitly defined, we assume that $J(A) = \emptyset$.

| Present State | Next State | |
|---------------|------------|---------|
| | $x = 0$ | $x = 1$ |
| A | A | D |
| B | A | C |
| C | C | B |
| D | C | A |

Table 2.2: State-transition table for finite-state machine M

| World | p | q | r | s |
|-------|-------|-------|-------|------|
| A | true | true | false | true |
| B | false | true | false | true |
| C | true | false | false | true |
| D | false | true | false | true |

Table 2.3: Truth values of primitive propositions p , q , r , and s in each world

The next example illustrates how a Kripke structure might be used to represent a state machine.

Example 2.9

Consider the state-transition table for a finite-state machine M shown in Table 2.2. This machine has four states: A , B , C , and D . The column labeled “Present State” lists the possible *present states* of M . The two columns under the label “Next State” list the *next states* of M if the input x is either 0 or 1, respectively. For example, the second row of Table 2.2 describes M ’s behavior whenever it is currently in state B : if the input is x is 0, then the next state will be A ; if x is 1, then the next state will be C .

We can construct a Kripke structure $\langle W_2, I_2, J_2 \rangle$ to model this machine by defining W_2 to be the set of M ’s states:

$$W_2 = \{A, B, C, D\}.$$

Now, suppose that there are four primitive propositions (p, q, r, s) associated with the state machine M , with their truth values in the various states given by Table 2.3. This table effectively specifies the interpretation function I_2 on these propositions, namely:

$$\begin{aligned} I_2(p) &= \{A, C\}, \\ I_2(q) &= \{A, B, D\}, \\ I_2(r) &= \{\}, \\ I_2(s) &= \{A, B, C, D\}. \end{aligned}$$

Finally, imagine that there is an observer Obs of the machine’s execution. This observer has faulty knowledge of M ’s states: whenever M is in state C , Obs incorrectly believes M to be in state D . We’ll assume that the observer *does* correctly know when M is in states A , B , or D .

This observer's state knowledge can be captured by the following relation:

$$J_2(Obs) = \{(A,A), (B,B), (C,D), (D,D)\}.$$

In the relation $J_2(Obs)$, the first element of each pair represents the present state of M , and the second element is the observed state of M . Thus the pair (C,D) reflects that, whenever M is in state C , Obs always believes the current state is D .

The tuple $\langle \{A,B,C,D\}, I_2, J_2 \rangle$ forms a Kripke structure. \diamond

In the next example, we model the same state machine, but we consider the inputs (i.e., “ $x=0$ ” or “ $x=1$ ”) as the “observers” of the state machine, and we use each “next state” as the perceived state by the particular observer. Although the set of worlds and the interpretation function do not change, the principal-mapping function does change.

Example 2.10

Let W_2 and I_2 be as defined in Example 2.9, and define J_3 as follows:

$$\begin{aligned} J_3(X_0) &= \{(A,A), (B,A), (C,C), (D,C)\}, \\ J_3(X_1) &= \{(A,D), (B,C), (C,B), (D,A)\}. \end{aligned}$$

The tuple $\langle \{A,B,C,D\}, I_2, J_3 \rangle$ forms a Kripke structure. \diamond

Just as the interpretation function I of a Kripke structure provides the base interpretation for propositional variables, the function J provides a base interpretation for simple principal names. We extend J to work over arbitrary *principal expressions*, using set union and relational composition as follows:

$$\begin{aligned} J(P \ \& \ Q) &= J(P) \cup J(Q), \\ J(P \mid Q) &= J(P) \circ J(Q). \end{aligned}$$


Example 2.11

Suppose that we have the following relations:

$$\begin{aligned} J(Andy) &= \{(w_0, w_0), (w_0, w_2), (w_1, w_1), (w_2, w_1)\}, \\ J(Stu) &= \{(w_1, w_2)\}, \\ J(Keri) &= \{(w_0, w_2), (w_1, w_2), (w_2, w_2)\}. \end{aligned}$$

Then $J(Keri \mid (Andy \ \& \ Stu))$ is calculated as follows:

$$\begin{aligned} J(Keri \mid (Andy \ \& \ Stu)) &= J(Keri) \circ J(Andy \ \& \ Stu), \\ &= J(Keri) \circ (J(Andy) \cup J(Stu)), \\ &= J(Keri) \circ \{(w_0, w_0), (w_0, w_2), (w_1, w_1), (w_2, w_1), (w_1, w_2)\} \\ &= \{(w_0, w_1), (w_1, w_1), (w_2, w_1)\}. \end{aligned} \quad \diamond$$

 **Exercise 2.3.1** Recall the Kripke structure $\langle W_0, I_0, J_0 \rangle$ from Example 2.7, and further suppose that

$$J_0(\text{Ida}) = \{(sw, sc), (sc, sw), (ns, sc), (ns, ns)\}.$$

Calculate the following relations:

- a. $J_0(\text{Hal} \ \& \ \text{Gil})$
- b. $J_0(\text{Gil} \mid \text{Hal})$
- c. $J_0(\text{Flo} \ \& \ \text{Ida})$
- d. $J_0(\text{Hal} \mid \text{Ida})$
- e. $J_0(\text{Ida} \mid \text{Hal})$
- f. $J_0(\text{Hal} \ \& \ (\text{Ida} \mid \text{Hal}))$
- g. $J_0(\text{Hal} \mid (\text{Ida} \ \& \ \text{Hal}))$

2.3.2 Semantics of the Logic

The Kripke structures provide the foundation for a formal, precise, and rigorous interpretation of formulas in our logic. For each Kripke structure $\mathcal{M} = \langle W, I, J \rangle$, we can define what it means for formulas in our logic to be *satisfied* in the structure. We can also identify those worlds in W for which a given formula is said to be true.

To define the semantics, we introduce a family of *evaluation functions*. Each Kripke structure $\mathcal{M} = \langle W, I, J \rangle$ gives rise to an evaluation function $\mathcal{E}_{\mathcal{M}}$ that maps well-formed formulas in the logic to subsets of W . Intuitively, $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$ is the set of worlds from the Kripke structure \mathcal{M} for which the well-formed formula φ is considered true. We say that \mathcal{M} *satisfies* φ (written $\mathcal{M} \models \varphi$) whenever φ is true in *all* of the worlds of \mathcal{M} : that is, when $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!] = W$. It follows that a Kripke structure \mathcal{M} *does not satisfy* φ (written $\mathcal{M} \not\models \varphi$) when there exists at least one $w \in W$ such that $w \notin \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$.

Each $\mathcal{E}_{\mathcal{M}}$ is defined inductively on the structure of well-formed formulas, making use of the interpretation functions I and J within the Kripke structure $\mathcal{M} = \langle W, I, J \rangle$. We discuss the individual cases separately, starting with the standard propositional operators and then moving on to the access-control specific cases. The full set of definitions is also summarized in Figure 2.1.

Standard Propositional Operators

The semantics for propositional variables and the standard logical connectives (e.g., negation, conjunction, implication) are very similar to the truth-table interpretations for standard propositional logic. The interpretation function I identifies those worlds in which the various propositional variables are true, while the semantics of the other operators are defined using standard set operations. We handle these cases in turn.

FIGURE 2.1 Semantics of core logic, for each $\mathcal{M} = \langle W, I, J \rangle$

$$\begin{aligned}
\mathcal{E}_{\mathcal{M}}[[p]] &= I(p) \\
\mathcal{E}_{\mathcal{M}}[[\neg\phi]] &= W - \mathcal{E}_{\mathcal{M}}[[\phi]] \\
\mathcal{E}_{\mathcal{M}}[[\phi_1 \wedge \phi_2]] &= \mathcal{E}_{\mathcal{M}}[[\phi_1]] \cap \mathcal{E}_{\mathcal{M}}[[\phi_2]] \\
\mathcal{E}_{\mathcal{M}}[[\phi_1 \vee \phi_2]] &= \mathcal{E}_{\mathcal{M}}[[\phi_1]] \cup \mathcal{E}_{\mathcal{M}}[[\phi_2]] \\
\mathcal{E}_{\mathcal{M}}[[\phi_1 \supset \phi_2]] &= (W - \mathcal{E}_{\mathcal{M}}[[\phi_1]]) \cup \mathcal{E}_{\mathcal{M}}[[\phi_2]] \\
\mathcal{E}_{\mathcal{M}}[[\phi_1 \equiv \phi_2]] &= \mathcal{E}_{\mathcal{M}}[[\phi_1 \supset \phi_2]] \cap \mathcal{E}_{\mathcal{M}}[[\phi_2 \supset \phi_1]] \\
\mathcal{E}_{\mathcal{M}}[[P \Rightarrow Q]] &= \begin{cases} W, & \text{if } J(Q) \subseteq J(P) \\ \emptyset, & \text{otherwise} \end{cases} \\
\mathcal{E}_{\mathcal{M}}[[P \text{ says } \phi]] &= \{w \mid J(P)(w) \subseteq \mathcal{E}_{\mathcal{M}}[[\phi]]\} \\
\mathcal{E}_{\mathcal{M}}[[P \text{ controls } \phi]] &= \mathcal{E}_{\mathcal{M}}[[P \text{ says } \phi \supset \phi]]
\end{aligned}$$

Propositional Variables: The truth of a propositional variable p is determined by the interpretation function I : a variable p is considered true in world w precisely when $w \in I(p)$. Thus, for all propositional variables p ,

$$\mathcal{E}_{\mathcal{M}}[[p]] = I(p).$$

For example, if \mathcal{M}_0 is the Kripke structure $\langle W_0, I_0, J_0 \rangle$ from Example 2.7, $\mathcal{E}_{\mathcal{M}_0}[[g]] = I_0(g) = \{sw\}$.

Negation: A formula with form $\neg\phi$ is true in precisely those worlds in which ϕ is *not* true. Because (by definition) $\mathcal{E}_{\mathcal{M}}[[\phi]]$ is the set of worlds in which ϕ is true, we define

$$\mathcal{E}_{\mathcal{M}}[[\neg\phi]] = W - \mathcal{E}_{\mathcal{M}}[[\phi]].$$

Thus, returning to Example 2.7,

$$\mathcal{E}_{\mathcal{M}_0}[[\neg g]] = W_0 - \mathcal{E}_{\mathcal{M}_0}[[g]] = \{sw, sc, ns\} - \{sw\} = \{sc, ns\}.$$

Notice that $\mathcal{E}_{\mathcal{M}_0}[[\neg g]]$ is the set of worlds in which the children are *not* allowed to go outside.

Conjunction: A conjunctive formula $\phi_1 \wedge \phi_2$ is considered true in those worlds for which *both* ϕ_1 and ϕ_2 are true: that is, $\phi_1 \wedge \phi_2$ is true in those worlds w for which $w \in \mathcal{E}_{\mathcal{M}}[[\phi_1]]$ and $w \in \mathcal{E}_{\mathcal{M}}[[\phi_2]]$. Thus, we can define $\mathcal{E}_{\mathcal{M}}[[\phi_1 \wedge \phi_2]]$ in terms of set intersection:

$$\mathcal{E}_{\mathcal{M}}[[\phi_1 \wedge \phi_2]] = \mathcal{E}_{\mathcal{M}}[[\phi_1]] \cap \mathcal{E}_{\mathcal{M}}[[\phi_2]].$$

Disjunction: Likewise, a disjunctive formula $\phi_1 \vee \phi_2$ is considered true in those worlds for which *at least one of* ϕ_1 and ϕ_2 is true: that is, $\phi_1 \vee \phi_2$ is true in those worlds w for which $w \in \mathcal{E}_{\mathcal{M}}[\![\phi_1]\!]$ or $w \in \mathcal{E}_{\mathcal{M}}[\![\phi_2]\!]$. Thus, we define $\mathcal{E}_{\mathcal{M}}[\![\phi_1 \vee \phi_2]\!]$ in terms of set union:

$$\mathcal{E}_{\mathcal{M}}[\![\phi_1 \vee \phi_2]\!] = \mathcal{E}_{\mathcal{M}}[\![\phi_1]\!] \cup \mathcal{E}_{\mathcal{M}}[\![\phi_2]\!].$$

Implication: An implication $\phi_1 \supset \phi_2$ is true in those worlds w for which either ϕ_2 is true (i.e., $w \in \mathcal{E}_{\mathcal{M}}[\![\phi_2]\!]$) or ϕ_1 is not true (i.e., $w \notin \mathcal{E}_{\mathcal{M}}[\![\phi_1]\!]$, and thus $w \in \mathcal{E}_{\mathcal{M}}[\![\neg\phi_1]\!]$). That is, $\phi_1 \supset \phi_2$ is true in those worlds in which, if ϕ_1 is true, then ϕ_2 is also true; if ϕ_1 is false, then ϕ_2 's interpretation is immaterial. Thus, we define the semantics of implications as follows:

$$\mathcal{E}_{\mathcal{M}}[\![\phi_1 \supset \phi_2]\!] = (W - \mathcal{E}_{\mathcal{M}}[\![\phi_1]\!]) \cup \mathcal{E}_{\mathcal{M}}[\![\phi_2]\!].$$

Equivalence: An equivalence $\phi_1 \equiv \phi_2$ is true in exactly those worlds w in which the implications $\phi_1 \supset \phi_2$ and $\phi_2 \supset \phi_1$ are *both* true. Thus, we define the semantics of implications as follows:

$$\mathcal{E}_{\mathcal{M}}[\![\phi_1 \equiv \phi_2]\!] = \mathcal{E}_{\mathcal{M}}[\![\phi_1 \supset \phi_2]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\phi_2 \supset \phi_1]\!].$$

Example 2.12

Let \mathcal{M}_1 be the Kripke structure $\langle W_1, I_1, J_1 \rangle$ from Example 2.8. The set $\mathcal{E}_{\mathcal{M}_1}[\![q \supset (r \wedge s)]\!]$ of worlds in W_1 in which the formula $q \supset (r \wedge s)$ is true is calculated as follows:

$$\begin{aligned} \mathcal{E}_{\mathcal{M}_1}[\![q \supset (r \wedge s)]\!] &= (W_1 - \mathcal{E}_{\mathcal{M}_1}[\![q]\!]) \cup \mathcal{E}_{\mathcal{M}_1}[\![r \wedge s]\!] \\ &= (W_1 - I_1(q)) \cup (\mathcal{E}_{\mathcal{M}_1}[\![r]\!] \cap \mathcal{E}_{\mathcal{M}_1}[\![s]\!]) \\ &= (W_1 - \{w_0, w_2\}) \cup (I_1(r) \cap I_1(s)) \\ &= \{w_1\} \cup (\{w_1\} \cap \{w_1, w_2\}) \\ &= \{w_1\} \cup \{w_1\} \\ &= \{w_1\}. \end{aligned}$$

◇

In the following example, we evaluate the same formula as in the previous example, but with respect to a different Kripke structure.

Example 2.13

Let \mathcal{M}_2 be the Kripke structure $\langle W_2, I_2, J_2 \rangle$ from Example 2.9. The set $\mathcal{E}_{\mathcal{M}_2} \llbracket q \supset (r \wedge s) \rrbracket$ of worlds W_2 in which the formula $q \supset (r \wedge s)$ is true is calculated as follows:

$$\begin{aligned}
 \mathcal{E}_{\mathcal{M}_2} \llbracket q \supset (r \wedge s) \rrbracket &= (W_2 - \mathcal{E}_{\mathcal{M}_2} \llbracket q \rrbracket) \cup \mathcal{E}_{\mathcal{M}_2} \llbracket r \wedge s \rrbracket \\
 &= (W_2 - I_2(q)) \cup (\mathcal{E}_{\mathcal{M}_2} \llbracket r \rrbracket \cap \mathcal{E}_{\mathcal{M}_2} \llbracket s \rrbracket) \\
 &= (W_2 - \{A, B, D\}) \cup (I_2(r) \cap I_2(s)) \\
 &= \{C\} \cup (\emptyset \cap \{A, B, C, D\}) \\
 &= \{C\} \cup \emptyset \\
 &= \{C\}.
 \end{aligned}$$

◇

Access-Control Operators

The access-control operators of the logic (e.g., *says*, *controls*, and \Rightarrow) have more interesting semantics.

Says: A formula P *says* ϕ is meant to denote a situation in which the principal P makes the statement ϕ . Intuitively, a principal should make statements that they *believe* to be true. What does it mean for a principal to believe a statement is true in a given world? The standard answer is that a principal P believes ϕ to be true in a specific world w if ϕ is true in all of the worlds w' that P *can conceive* the current world to be (i.e., all w' such that (w, w') is in $J(P)$). Of course, this set of *possible worlds* is simply the set $J(P)(w)$; ϕ is true in every world in $J(P)(w)$ if and only if $J(P)(w) \subseteq \mathcal{E}_{\mathcal{M}} \llbracket \phi \rrbracket$. Therefore, we define

$$\mathcal{E}_{\mathcal{M}} \llbracket P \text{ says } \phi \rrbracket = \{w \mid J(P)(w) \subseteq \mathcal{E}_{\mathcal{M}} \llbracket \phi \rrbracket\}.$$

Controls: Formulas of the form P *controls* ϕ express a principal P 's jurisdiction or authority regarding the statement ϕ . We interpret P *controls* ϕ as syntactic sugar for the statement $(P \text{ says } \phi) \supset \phi$, which captures the desired intuition: if the authority P says that ϕ is true, then ϕ is true. Thus, we give the meaning of P *controls* ϕ directly as the meaning of this rewriting:

$$\mathcal{E}_{\mathcal{M}} \llbracket P \text{ controls } \phi \rrbracket = \mathcal{E}_{\mathcal{M}} \llbracket (P \text{ says } \phi) \supset \phi \rrbracket.$$

Speaks For: To understand the semantics of formulas with form $P \Rightarrow Q$, recall the purpose of such formulas: we wish to express a proxy relationship between P and Q that will permit us to safely attribute P 's statements to Q as well, independent of a particular world. That is, if $P \Rightarrow Q$, then it should be reasonable to interpret any statement from P as being a statement that Q would also make. In terms of the semantics, we have seen that a principal P making a statement ϕ in a world w means that $J(P)(w) \subseteq \mathcal{E}_{\mathcal{M}} \llbracket \phi \rrbracket$. Thus, if we wish to associate *all* of P 's statements to Q , then we need to know that $J(Q)(w) \subseteq J(P)(w)$ for

all worlds w . If $J(Q) \subseteq J(P)$, then this relationship naturally holds. Therefore, we define

$$\mathcal{E}_{\mathcal{M}}[P \Rightarrow Q] = \begin{cases} W, & \text{if } J(Q) \subseteq J(P) \\ \emptyset, & \text{otherwise.} \end{cases}$$

The following examples illustrate these semantic definitions.

Example 2.14

Recall $\mathcal{M}_0 = \langle W_0, I_0, J_0 \rangle$ from Example 2.7. The set of worlds in W_0 in which the formula *Hal* says g is true is given by $\mathcal{E}_{\mathcal{M}_0}[\llbracket \text{Hal says } g \rrbracket]$, which is calculated as follows:

$$\begin{aligned} \mathcal{E}_{\mathcal{M}_0}[\llbracket \text{Hal says } g \rrbracket] &= \{w \mid J_0(\text{Hal})(w) \subseteq \mathcal{E}_{\mathcal{M}_0}[\llbracket g \rrbracket]\} \\ &= \{w \mid J_0(\text{Hal})(w) \subseteq \{sw\}\} \\ &= \{sw, sc\}. \end{aligned}$$

This result captures *Hal*'s mistaken belief that, whenever it is sunny (i.e., when the current world is either sw or sc), the children will be able to go outside.

In contrast, recall that *Flo* is unable to distinguish the two worlds sw and sc . Specifically, the relation $J_0(\text{Flo})$ has the following properties:

$$\begin{aligned} J_0(\text{Flo})(sw) &= \{sw, sc\}, \\ J_0(\text{Flo})(sc) &= \{sw, sc\}, \\ J_0(\text{Flo})(ns) &= \{ns\}. \end{aligned}$$

Thus, the worlds in which *Flo* says g is true can be calculated as follows:

$$\begin{aligned} \mathcal{E}_{\mathcal{M}_0}[\llbracket \text{Flo says } g \rrbracket] &= \{w \mid J_0(\text{Flo})(w) \subseteq \mathcal{E}_{\mathcal{M}_0}[\llbracket g \rrbracket]\} \\ &= \{w \mid J_0(\text{Flo})(w) \subseteq \{sw\}\} \\ &= \emptyset. \end{aligned}$$

That is, there are no worlds in which *Flo* is convinced that the children will be able to go outside. \diamond

Example 2.15

Recall $\mathcal{M}_1 = \langle W_1, I_1, J_1 \rangle$ from Examples 2.8 and Example 2.12. The set of worlds in W_1 in which the formula *Alice* says $(q \supset (r \wedge s))$ is true is given by $\mathcal{E}_{\mathcal{M}_1}[\llbracket \text{Alice says } (q \supset (r \wedge s)) \rrbracket]$, calculated as follows:

$$\begin{aligned} \mathcal{E}_{\mathcal{M}_1}[\llbracket \text{Alice says } (q \supset (r \wedge s)) \rrbracket] &= \{w \mid J_1(\text{Alice})(w) \subseteq \mathcal{E}_{\mathcal{M}_1}[\llbracket q \supset (r \wedge s) \rrbracket]\} \\ &= \{w \mid J_1(\text{Alice})(w) \subseteq \{w_1\}\} \\ &= \{w_1\}. \end{aligned}$$

This result is not surprising, because *Alice* had perfect knowledge of the separate worlds: thus, she believes $q \supset (r \wedge s)$ to be true in precisely those worlds in which it is true. \diamond

Example 2.16

Consider the same Kripke structure $\mathcal{M}_1 = \langle W_1, I_1, J_1 \rangle$ as in the previous example, but now we investigate *Bob's* view of the situation. Recall the following details about $J_1(\text{Bob})$:

$$\begin{aligned} J_1(\text{Bob})(w_0) &= \{w_0, w_1\}, \\ J_1(\text{Bob})(w_1) &= \{w_2\}, \\ J_1(\text{Bob})(w_2) &= \{w_1\}. \end{aligned}$$

The set of worlds in W_1 in which the formula *Bob* says $(q \supset (r \wedge s))$ is true is given by $\mathcal{E}_{\mathcal{M}_1}[\text{Bob says } (q \supset (r \wedge s))]$, which is calculated as follows:

$$\begin{aligned} \mathcal{E}_{\mathcal{M}_1}[\text{Bob says } (q \supset (r \wedge s))] &= \{w \mid J_1(\text{Bob})(w) \subseteq \mathcal{E}_{\mathcal{M}_1}[q \supset (r \wedge s)]\} \\ &= \{w \mid J_1(\text{Bob})(w) \subseteq \{w_1\}\} \\ &= \{w_2\}. \end{aligned} \quad \diamond$$

As remarked at the beginning of this section, a particular Kripke structure $\mathcal{M} = \langle W, I, J \rangle$ satisfies a formula ϕ , written $\mathcal{M} \models \phi$, if and only if ϕ is true of every world in W (i.e., if $\mathcal{E}_{\mathcal{M}}[\phi] = W$). The following examples illustrate this concept.

Example 2.17

Recall the Kripke structure $\mathcal{M}_1 = \langle W_1, I_1, J_1 \rangle$ from Examples 2.8 and 2.12. \mathcal{M}_1 satisfies the formula $q \vee r$, but not the formula $q \supset (r \wedge s)$:

$$\begin{aligned} \mathcal{E}_{\mathcal{M}_1}[q \vee r] &= \{w_0, w_1, w_2\} = W_1, \\ \mathcal{E}_{\mathcal{M}_1}[q \supset (r \wedge s)] &= \{w_1\} \neq W_1. \end{aligned}$$

Thus, we write $\mathcal{M}_1 \models q \vee r$ and $\mathcal{M}_1 \not\models q \supset (r \wedge s)$. \diamond


Example 2.18

The same Kripke structure $\mathcal{M}_1 = \langle W_1, I_1, J_1 \rangle$ from Examples 2.8 and 2.12 also satisfies the formula *Alice* controls $(q \supset (r \wedge s))$, because:

$$\begin{aligned} \mathcal{E}_{\mathcal{M}_1}[\text{Alice controls } (q \supset (r \wedge s))] &= \mathcal{E}_{\mathcal{M}_1}[\text{Alice says } (q \supset (r \wedge s)) \supset (q \supset (r \wedge s))] \\ &= (W_1 - \mathcal{E}_{\mathcal{M}_1}[\text{Alice says } (q \supset (r \wedge s))]) \cup \mathcal{E}_{\mathcal{M}_1}[q \supset (r \wedge s)] \\ &= (W_1 - \{w_1\}) \cup \{w_1\} \\ &= W_1. \end{aligned}$$

Therefore, we can write $\mathcal{M}_1 \models \text{Alice controls } (q \supset (r \wedge s))$. \diamond


The following exercises provide opportunities to use the Kripke semantics to calculate the meanings of access-control formulas, to identify structures that satisfy given formulas, and to prove useful properties about the semantics.

 **Exercise 2.3.2** Suppose \mathcal{M} is the Kripke structure $\langle W, I, J \rangle$, where W , I , and J are defined as follows:

$$\begin{aligned} W &= \{w_0, w_1, w_2\} \\ I(s) &= \{w_1, w_2\} \\ I(t) &= \{w_2\} \\ J(Cy) &= \{(w_1, w_0), (w_1, w_1), (w_2, w_0)\} \\ J(Di) &= \{(w_0, w_1), (w_1, w_0), (w_2, w_2)\}. \end{aligned}$$

For each of the following formulas, give the set of worlds in W for which the formula is true (i.e., calculate $\mathcal{E}_{\mathcal{M}}[\![\Phi]\!]$ for each formula Φ).

- a. $s \supset t$
- b. $\neg(s \supset t)$
- c. $Cy \text{ says } (s \supset t)$
- d. $Cy \text{ says } \neg(s \supset t)$
- e. $Di \text{ says } (s \supset t)$
- f. $Di \text{ says } \neg(s \supset t)$
- g. $(Cy \ \& \ Di) \text{ says } (s \supset t)$
- h. $(Cy \ \& \ Di) \text{ says } \neg(s \supset t)$
- i. $(Di \mid Cy) \text{ says } (s \supset t)$
- j. $(Di \mid Cy) \text{ says } \neg(s \supset t)$
- k. $Di \Rightarrow Cy$
- l. $Cy \text{ says } (Di \Rightarrow Cy)$
- m. $Di \text{ says } (Di \Rightarrow Cy)$
- n. $Di \text{ says } (Di \ \& \ Cy \Rightarrow Cy)$

 **Exercise 2.3.3** Let \mathcal{M} be the Kripke structure $\langle W, I, J \rangle$, where W , I , and J are defined as follows:

- $W = \{t, u, v, x, y, z\}$

- $I : \text{PropVar} \rightarrow 2^W$ given by:

$$\begin{aligned} I(p) &= \{x, y, z\} \\ I(q) &= \{x, y, t\} \\ I(r) &= \{y, t, u, z\} \end{aligned}$$

- $J : \text{PName} \rightarrow 2^{W \times W}$ given by:

$$\begin{aligned} J(A) &= \{(w, w) \mid w \in W\} \cup \{(x, y), (x, z), (z, t), (y, v), (v, y), (v, x)\} \\ J(B) &= \{(x, w) \mid w \in W\} \cup \{(y, t), (z, t), (t, v)\}. \end{aligned}$$

Calculate each of the following sets.

- $\mathcal{E}_{\mathcal{M}}[(p \supset q) \supset r]$
- $\mathcal{E}_{\mathcal{M}}[A \text{ says } (p \supset r)]$
- $\mathcal{E}_{\mathcal{M}}[A \text{ says } (B \text{ says } q)]$
- $\mathcal{E}_{\mathcal{M}}[B \text{ says } (B \text{ says } q)]$
- $\mathcal{E}_{\mathcal{M}}[A \text{ controls } (B \text{ says } q)]$
- $\mathcal{E}_{\mathcal{M}}[A \text{ controls } (B \text{ controls } q)]$



Exercise 2.3.4 Let \mathcal{M} be the Kripke structure $\langle W, I, J \rangle$, where

$$\begin{aligned} W &= \{n, r, c, d, rc, rd, cd, rcd\} \\ I(\text{read}) &= \{r, rc, rd, rcd\} \\ I(\text{copy}) &= \{c, rc, cd, rcd\} \\ I(\text{del}) &= \{d, rd, cd, rcd\} \end{aligned}$$

$$\begin{aligned} J(X) &= \{(w, rcd) \mid w \in W\} \\ J(Y) &= \text{id}_W \\ J(Z) &= \{(w, n) \mid w \in W\}. \end{aligned}$$

- Prove that $\mathcal{M} \models Y \text{ controls } (\text{read} \wedge \text{copy})$.
- Prove that $\mathcal{M} \not\models X \text{ controls } \text{del}$.



Exercise 2.3.5 Let $\mathcal{A} = \langle W_A, I_A, J_A \rangle$ and $\mathcal{B} = \langle W_B, I_B, J_B \rangle$ be the Kripke structures

defined by:

$$\begin{aligned}
 W_A &= \{a_1, a_2, a_3\} \\
 I_A(p) &= \{a_3\} \\
 I_A(q) &= \{a_1, a_3\} \\
 I_A(r) &= \{a_1, a_2\} \\
 J_A(Val) &= \{(a_1, a_1), (a_1, a_2), (a_2, a_1), (a_3, a_2)\} \\
 J_A(Wyn) &= \{(a_1, a_3), (a_2, a_3), (a_3, a_2)\}.
 \end{aligned}$$

$$\begin{aligned}
 W_B &= \{b_1, b_2, b_3, b_4\} \\
 I_B(p) &= \{b_2, b_1\} \\
 I_B(q) &= \{b_3\} \\
 I_B(r) &= \{b_2, b_4, b_1\} \\
 J_B(Val) &= \{(b_1, b_2), (b_2, b_3), (b_2, b_4), (b_4, b_4)\} \\
 J_B(Wyn) &= \{(b_1, b_1), (b_2, b_1), (b_3, b_2), (b_3, b_4), (b_4, b_2)\}.
 \end{aligned}$$

Which of the following statements are true? Support your answers with calculations using the evaluation functions $\mathcal{E}_A[\![\!-\!]\!]$ and $\mathcal{E}_B[\![\!-\!]\!]$.

- a. $\mathcal{A} \models p \vee (q \supset r)$
- b. $\mathcal{B} \models p \vee (q \supset r)$
- c. $\mathcal{A} \models Val \Rightarrow Wyn$
- d. $\mathcal{B} \models Val \Rightarrow Wyn$
- e. $\mathcal{A} \models Val \mid Wyn \text{ says } r$
- f. $\mathcal{B} \models Val \mid Wyn \text{ says } r$
- g. $\mathcal{A} \models Wyn \text{ controls } (p \wedge r)$
- h. $\mathcal{B} \models Wyn \text{ controls } (p \wedge r)$



Exercise 2.3.6 For each of the following formulas ϕ , find Kripke structures \mathcal{M}_x and \mathcal{M}_y such that $\mathcal{M}_x \models \phi$ and $\mathcal{M}_y \not\models \phi$.

- a. Ned says $(p \equiv q)$
- b. Olaf controls q
- c. Pam says $(Rue \text{ controls } r)$
- d. Sal controls $(Tom \Rightarrow Ugo)$



Exercise 2.3.7 Prove that, for any Kripke structure $\mathcal{M} = \langle W, I, J \rangle$, principal P , and formulas φ_1 and φ_2 , the following relationship holds:

$$\mathcal{E}_{\mathcal{M}}[[P \text{ says } (\varphi_1 \equiv \varphi_2)]] \cap \mathcal{E}_{\mathcal{M}}[[P \text{ says } \varphi_1]] \subseteq \mathcal{E}_{\mathcal{M}}[[P \text{ says } \varphi_2]].$$



Exercise 2.3.8 Prove that, for every Kripke structure $\mathcal{M} = \langle W, I, J \rangle$, principal P , and formulas φ_1 and φ_2 , the following relationship holds:

$$\mathcal{M} \models P \text{ says } (\varphi_1 \equiv \varphi_2) \supset (P \text{ says } \varphi_1 \supset P \text{ says } \varphi_2).$$

Hint: Exercise 2.3.7 is helpful here.



Exercise 2.3.9 Prove that, for any Kripke structure $\mathcal{M} = \langle W, I, J \rangle$ and formulas φ_1 and φ_2 , $\mathcal{E}_{\mathcal{M}}[[\varphi_1 \equiv \varphi_2]] = W$ if and only if $\mathcal{E}_{\mathcal{M}}[[\varphi_1]] = \mathcal{E}_{\mathcal{M}}[[\varphi_2]]$. That is, prove the following two statements:

- a. If $\mathcal{E}_{\mathcal{M}}[[\varphi_1 \equiv \varphi_2]] = W$, then $\mathcal{E}_{\mathcal{M}}[[\varphi_1]] = \mathcal{E}_{\mathcal{M}}[[\varphi_2]]$.
- b. If $\mathcal{E}_{\mathcal{M}}[[\varphi_1]] = \mathcal{E}_{\mathcal{M}}[[\varphi_2]]$, then $\mathcal{E}_{\mathcal{M}}[[\varphi_1 \equiv \varphi_2]] = W$.

2.4 Summary

Providing and maintaining security—in both the physical and digital worlds—requires us to be able to determine whether or not any given decision to grant access to resources or services is correct. Unfortunately, natural language expressions have imprecise meanings. We need a way to unambiguously express the policies, trust assumptions, recognized authorities, and statements made by various principals and be able to justify the resulting access-control decisions.

In this chapter, we introduced a language that allows us to express our policies, trust assumptions, recognized authorities, and statements in a precise and unambiguous way. Expressions in this language are given precise, mathematical meanings through the use of Kripke structures. This Kripke semantics provides the initial basis for mathematically justifying access-control decisions: given a Kripke structure and an expression in the language, we can compute those worlds in which the expression is true. This semantics will allow us to introduce sound rules for justifying access-control decisions, as demonstrated in the next chapter.

The learning outcomes associated with this chapter appear in Figure 2.2.

2.5 Further Reading

For a detailed introduction to discrete mathematics and propositional logic, we direct the interested reader to some of the standard undergraduate textbooks on the

FIGURE 2.2 Learning outcomes for Chapter 2

After completing this chapter, you should be able to achieve the following learning outcomes at several levels of knowledge:

Application

- Determine whether or not a given collection of symbols constitutes a well-formed formula of the access-control logic.
- When given an informal statement in English, translate it into a well-formed formula of the access-control logic.
- When given a Kripke structure and a formula of the access-control logic, determine the set of worlds in which the formula is true.

Analysis

- When given a formula and a Kripke structure that satisfies it, prove that the Kripke structure satisfies it.

Synthesis

- When given a satisfiable formula, construct a Kripke structure that satisfies the formula.
- When given a refutable formula, construct a Kripke structure that refutes (i.e., fails to satisfy) the formula.
- When given a general property regarding Kripke structures, prove that the property holds.

Evaluation

- When given a Kripke structure and a formula, determine whether or not the Kripke structure satisfies the formula.
-

subject (Rosen, 2003; Ross and Wright, 2002).

The access-control logic presented here is based on the logic of Abadi, Lampson, and colleagues (Lampson et al., 1992; Abadi et al., 1993), which in turn was built on top of a standard modal logic. Modal logic is the logical study of necessity and possibility, and Hughes and Cresswell provide an excellent introduction to the field (Hughes and Cresswell, 1996).