Chapter 9

Credentials-based Authorization

A set can be defined *intensionally* by specifying properties required of all its members or it can be defined *extensionally* by enumerating its elements. For example, the set of people authorized to enter a nightclub might be characterized intensionally by giving a minimum required age or characterized extensionally by providing a guest list. The DAC and MAC authorization policies we have been studying enumerate principals (with privileges), so they are extensionally-defined policies. As a result, these authorization policies do not provide a useful explanation about why a given request is or is not authorized.

An intensionally-defined authorization policy would supply such an explanation because, by definition, authorization is decided by checking whether certain properties are satisfied. The properties that needed to be satisfied but aren't constitute the explanation for why a given request is not authorized.

Properties on which we might base an authorization decision include

- beliefs about principals,
- beliefs about other aspects of the system state, and
- the basis for trusting each of these beliefs.

Cornell University, for example, might stipulate that a request to read from the university's telephone directory be granted only if made during normal working hours by an individual who Cornell certifies as being among its students, staff, or faculty. This is an intensionally-defined policy. It is formulated in terms of a belief about the system state ("during normal working hours"), a belief about the principal making the request ("being among the students, ..."), and a basis for trusting the latter belief ("... who Cornell certifies").

Credentials-based authorization[1] uses *credentials* and *guards* to enforce inten-

[1]This is also called *claims-based authorization* and *proof-carrying authorization* in the literature.

sionally-defined authorization policies. *Credentials* convey beliefs about principals and/or the system state; *guards* employ logical inference to grant requests only when some specified *goal formula* is shown to hold. The goal formula is a logical formula involving (i) predicates characterizing beliefs that principals hold and (ii) predicates characterizing which *sources* of credentials the guard trusts to convey accurate beliefs about one or another aspect of reality. A guard's decision to grant a request is thus based on properties (described in a goal formula and conveyed by credentials), which is the defining characteristic of an intensionally-defined authorization policy.

Note that guards in credentials-based authorization do not themselves check whether beliefs conveyed in credentials are accurate statements about reality. What a guard checks is:

- Whether a goal formula is satisfied by beliefs being conveyed in some assembled credentials.

- Whether the sources of those credentials are trusted by this guard.

The source of each credential is the accountable party for ensuring the accuracy of beliefs conveyed by that credential.

Credentials-based authorization allows for *delegation of authority* because it enables different principals to be trusted on different matters. For example, a professional society is the authority on its membership, and a university is the authority on who are its students; with credentials-based authorization, both institutions would participate in enforcing an authorization policy for granting reduced conference registration fees to student members of the society. Delegation of authority is an attractive way to handle authorization in networked systems, where different hosts are managed by different organizations and thus each host can provide only some of the information needed to make an authorization decision. The alternative to delegation of authority is for guards to be responsible for storing and managing information. That approach is inferior because it replicates information already being stored and managed elsewhere in a system and because the guard now incorporates additional mechanism, which must be trusted.

9.1 A Logic for Authorization

The foundation of any realization of credentials-based authorization is a logic for specifying and reasoning about beliefs and goal formulas. This logic must be able to distinguish between identical beliefs held by different principals; it also must be able to distinguish between beliefs a principal holds and what is actually *true*. One principal might, for example, hold a belief that B is *true* while another disagrees or is ignorant about this matter; and a principal might even hold a belief that B is *true* when, in fact, B is *false*. First-order predicate logics, familiar to most computer scientists, do not provide convenient constructs for making such distinctions.

Modal logics offer a syntax for associating beliefs with principals, and these logics also provide inference rules that take into account the principal that is holding each belief involved in instantiating an inference rule. In this chapter, we describe a simple modal logic called CAL ($\underline{\text{C}}$redentials $\underline{\text{A}}$uthorization $\underline{\text{L}}$ogic), which is formulated expressly to serve as the foundation for credentials-based authorization schemes.

CAL extends a constructive first-order predicate logic CFoL by adding a **says** operator for attribution of beliefs and **speaksfor** operators for delegation.

- P **says** $\mathcal{C}$ attributes a belief—specified here by CAL formula $\mathcal{C}$—to a principal P. This construct is useful in formalizing access requests and meanings of credentials.
- P' **speaksfor** P asserts that a principal P adopts all beliefs attributed to principal P'. P' **says** $\mathcal{C}$ then implies P **says** $\mathcal{C}$. Unrestricted delegation from P to P' is specified in this manner, as is complete trust in P' by P.
- P' **speaks** $x{:}\mathcal{C}$ **for** P specifies *restricted delegation.* It asserts that certain beliefs—denoted here using notation "$x{:}\mathcal{C}$" (defined on page 215)—attributed to a principal P' are adopted by principal P.

These new operators enable an authorization policy to be specified as a CAL formula. For instance, CAL formula

$$\begin{array}{ll} & \mathit{Alice}\ \textbf{says}\ \mathit{PhoneNum}(\mathit{nme}) \\ \wedge & \mathit{TimeServer}\ \textbf{says}\ 0800 \leq \mathit{now} \leq 1700 \\ \wedge & \mathit{Cornell}\ \textbf{says}\ \mathit{Alice} \in (\mathit{CUstudents} \cup \mathit{CUstudents} \cup \mathit{CUstaff}) \end{array}$$

could serve as a goal formula for authorizing a *PhoneNum*(*nme*) request by *Alice* for the phone number of *nme*. It authorizes requests made during business hours (0800 to 1700) by a members of the Cornell community.

Given a goal formula specified in CAL, the task of a guard can then be characterized in terms of CAL inferences. For formulas $\mathcal{F}_1$, ..., $\mathcal{F}_n$ and $\mathcal{F}$ of any logic L, logicians write *sequent*

$$\mathcal{F}_1,\ \mathcal{F}_2,\ ...,\ \mathcal{F}_n \vdash_{\mathrm{L}} \mathcal{F} \tag{9.1}$$

to assert that *conclusion* $\mathcal{F}$ can be derived from *assumptions* $\mathcal{F}_1$, $\mathcal{F}_2$, ..., $\mathcal{F}_n$ by using the axioms and inference rules of logic L. We refer to such a derivation as *support* for the sequent. For logics that are sound, having support for a sequent implies that the sequent's conclusion is satisfied whenever the sequent's assumptions are satisfied. Notice that when the list of assumptions in (9.1) is empty then it becomes $\vdash_{\mathrm{L}} \mathcal{F}$, the conventional notation for asserting that $\mathcal{F}$ is a *theorem* of logic L. That meaning agrees with the familiar definition for a theorem—a formula derived by starting with axioms and applying a finite sequence of inference rules.

The operation of a guard thus can be characterized in terms of CAL and sequents, as follows.

Guard Operation. In response to each request R, a guard will:

1. *Goal Formula Determination.* Identify CAL formula $\mathcal{G}_R$ that should serve as the goal formula for request R.
2. *Credential Collection.* Assemble a collection of credentials $\mathtt{C}_1$, $\mathtt{C}_2$, ..., $\mathtt{C}_n$, for use in establishing that $\mathcal{G}_R$ holds.
3. *Guard Sequent Formulation.* Formulate a *guard sequent*
$$\mathcal{M}(\mathtt{C}_1),\ \mathcal{M}(\mathtt{C}_2),\ \ldots,\ \mathcal{M}(\mathtt{C}_n) \vdash_{\mathrm{CAL}} \mathcal{G}_R \tag{9.2}$$
where $\mathcal{M}(\mathtt{C})$ is the CAL formula denoting the belief(s) that credential $\mathtt{C}$ conveys.
4. *Authorization Decision.* Authorize request R to proceed if CAL support for (9.2) is available; deny R if that support is absent. □

This description admits many possible guard implementations. Credentials assembled in step 2 might accompany request R, be provided by a third party, and/or be fetched by the guard either in anticipation of R or only after R has been received. Similarly, the support for guard sequent (9.2) required by step 4 might accompany the request, be provided by a third party, or be generated by the guard.

Notice that the credentials in step 2, guard sequent in step 3, and CAL support in step 4 together constitute a rationale that can be understood by humans, can be recorded for later review, and identifies sources to hold accountable for each credential involved in the authorization decision.

9.2 A Constructive First-Order Predicate Logic

Formulas and Derivation Trees. The syntax for formulas $\mathcal{F}$ of the constructive first-order predicate logic CFoL that CAL extends is given by the BNF grammar

$$\begin{aligned}\mathcal{F} ::= \mathit{true} \ \mid\ \mathit{false} \ \mid\ p(\tau_1,\ \tau_2,\ \ldots,\ \tau_n)\\ \mid\ \mathcal{F}\wedge\mathcal{F}\ \mid\ \mathcal{F}\vee\mathcal{F}\ \mid\ \mathcal{F}\Rightarrow\mathcal{F}\\ \mid\ (\forall v\colon \mathcal{F})\ \mid\ (\exists v\colon \mathcal{F})\end{aligned} \tag{9.3}$$

where τ_1, τ_2, ..., τ_n are *terms* (i.e., expressions that map states to values) and p names a *predicate* (i.e., a total function that maps states to Booleans). In this logic, a predicate that takes zero arguments is called a *propositional variable*, and negation is considered an abbreviation:

$$\neg\mathcal{F}\colon\ (\mathcal{F}\Rightarrow \mathit{false}).$$

Figure 9.1 gives the inference rules of CFoL. Notation

$$\text{R:}\frac{\mathcal{P}_1,\ \mathcal{P}_2,\ \ldots,\ \mathcal{P}_n}{\mathcal{F}} \tag{9.4}$$

$$\text{TRUE: } \frac{}{\mathit{true}} \qquad\qquad \text{FALSE: } \frac{\mathit{false}}{\mathcal{F}}$$

$$\text{AND-I: } \frac{\mathcal{F}\ ,\ \mathcal{G}}{\mathcal{F}\wedge\mathcal{G}} \qquad \text{AND-LEFT-E: } \frac{\mathcal{F}\wedge\mathcal{G}}{\mathcal{F}} \qquad \text{AND-RIGHT-E: } \frac{\mathcal{F}\wedge\mathcal{G}}{\mathcal{G}}$$

$$\text{OR-LEFT-I: } \frac{\mathcal{F}}{\mathcal{F}\vee\mathcal{G}} \qquad \text{OR-RIGHT-I: } \frac{\mathcal{G}}{\mathcal{F}\vee\mathcal{G}} \qquad \text{OR-E: } \frac{\mathcal{F}\Rightarrow\mathcal{H}\ ,\ \mathcal{G}\Rightarrow\mathcal{H}\ ,\ \mathcal{F}\vee\mathcal{G}}{\mathcal{H}}$$

$$\text{IMP-E: } \frac{\mathcal{F}\ ,\ \mathcal{F}\Rightarrow\mathcal{G}}{\mathcal{G}} \qquad\qquad \text{IMP-I}(\lambda)\text{: } \frac{\begin{array}{c}\underline{\lambda\colon \mathcal{F}}\\ \vdots\\ \hline \mathcal{G}\end{array}}{\mathcal{F}\Rightarrow\mathcal{G}}$$

$$\text{FORALL-I: } \frac{\mathcal{F}}{(\forall x\colon \mathcal{F})} \qquad \begin{array}{l}\text{provided } x \text{ not free in any uncanceled}\\ \text{assumptions in the derivation of } \mathcal{F}\end{array}$$

$$\text{FORALL-E: } \frac{(\forall x\colon \mathcal{F})}{\mathcal{F}[x := \tau]} \qquad \begin{array}{l}\text{provided free variables in } \tau \text{ remain free}\\ \text{occurrences when } \tau \text{ is substituted for } x \text{ in } \mathcal{F}\end{array}$$

$$\text{EXISTS-I: } \frac{\mathcal{F}[x := \tau]}{(\exists x\colon \mathcal{F})} \qquad \begin{array}{l}\text{provided free variables in } \tau \text{ remain free}\\ \text{occurrences when } \tau \text{ is substituted for } x \text{ in } \mathcal{F}\end{array}$$

$$\text{EXISTS-E: } \frac{\mathcal{F}\Rightarrow\mathcal{G}\ ,\ (\exists x\colon \mathcal{F})}{\mathcal{G}} \qquad \begin{array}{l}\text{provided } x \text{ is not free in } \mathcal{G} \text{ or in any}\\ \text{uncanceled assumptions in the}\\ \text{derivation of } \mathcal{F}\Rightarrow\mathcal{G}\end{array}$$

Figure 9.1: Inference Rules for CFoL

is used there to specify an inference rule that has name R and that derives *conclusion* $\mathcal{F}$ from *premises* $\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_n$.

A *derivation tree* is a diagram that depicts how instances of inference rules are combined to derive a specific *conclusion* from some set of *assumptions.* Figure 9.2 gives an example. The conclusion (i.e., "$p \wedge q \Rightarrow q \wedge p$") appears at the bottom of the diagram and the assumptions appear at the top. A $\boxed{\text{box}}$ signifies an assumption that is deemed *canceled* because (i) it is an axiom or previously proved theorem, or (ii) the assumption has some label λ and the path in the derivation tree from that assumption to the conclusion includes

$$\text{IMP-I}(\lambda):\dfrac{\text{AND-I}:\dfrac{\text{AND-RIGHT-E}:\dfrac{\boxed{\lambda:\ p\wedge q}}{q},\quad \text{AND-LEFT-E}:\dfrac{\boxed{\lambda:\ p\wedge q}}{p}}{q\wedge p}}{p\wedge q\Rightarrow q\wedge p}$$

Figure 9.2: Derivation Tree for $p \wedge q \Rightarrow q \wedge p$

an application of inference rule IMP-I(λ). All other assumptions are deemed *uncanceled* and typeset without boxes. So the derivation tree in Figure 9.2 has two canceled assumptions.[2] Both canceled assumptions are the same (formula "$p \wedge q$" with label λ) and each assumption was deemed canceled because the path from it to the conclusion passes through an application of inference rule IMP-I(λ).

We formally define a derivation tree $\mathcal{T}$, its set $\mathit{Asmpts}(\mathcal{T})$ of assumptions and its conclusion $\mathit{Conc}(\mathcal{T})$, inductively.

Derivation Tree Formal Definition.

- A formula $\mathcal{F}$ or a labeled formula λ:$\mathcal{F}$ standing alone constitutes a derivation tree $\mathcal{T}$ with $\mathit{Asmpts}(\mathcal{T}) = \{\mathcal{F}\}$ and $\mathit{Conc}(\mathcal{T}) = \mathcal{F}$.
- If $\mathcal{D}_1$, $\mathcal{D}_2$, ..., $\mathcal{D}_n$ are derivation trees then

$$\text{R}:\dfrac{\mathcal{D}_1,\ \mathcal{D}_2,\ ...,\ \mathcal{D}_n}{\mathcal{F}}$$

  is a derivation tree $\mathcal{T}$ with

$$\begin{aligned}\mathit{Asmpts}(\mathcal{T}) &= \cup_{1\le i\le n}\mathit{Asmpts}(\mathcal{D}_i)\\ \mathit{Conc}(\mathcal{T}) &= \mathcal{F}\end{aligned}$$

  provided each $\mathcal{D}_i$ *discharges* corresponding premise $\mathcal{P}_i$ of inference rule $\text{R}:\dfrac{\mathcal{P}_1,\ \mathcal{P}_2,\ ...,\ \mathcal{P}_n}{\mathcal{F}}$ □

Whether a derivation tree $\mathcal{D}_i$ discharges a premise $\mathcal{P}_i$ of some inference rule depends on the premise. Two kinds of premises are found in the inference rules of Figure 9.1. (Calligraphic identifiers $\mathcal{F}$, $\mathcal{G}$, and $\mathcal{H}$ here denote formulas of the logic.)

- If premise $\mathcal{P}_i$ is simply a formula then derivation tree $\mathcal{D}_i$ discharges $\mathcal{P}_i$ if $\mathit{Conc}(\mathcal{D}_i) = \mathcal{P}_i$.

[2]Don't be misled by Figure 9.2. Not all assumptions in a derivation tree will necessarily be deemed canceled, and not all canceled assumptions will have the same label (e.g., λ). Derivation tree (9.5) on page 209, for example, has no canceled assumptions. And each of the assumptions in derivation tree (9.25) on page 226 has a different label.

- If premise $\mathcal{P}_i$ is $\dfrac{\overline{\lambda\colon \mathcal{F}}}{\underline{\vdots}}\atop{\mathcal{G}}$ then derivation tree $\mathcal{D}_i$ discharges $\mathcal{P}_i$ if $\mathcal{F} \in \mathit{Asmpts}(\mathcal{D}_i)$ and $\mathit{Conc}(\mathcal{D}_i) = \mathcal{G}$.

For example, in the derivation tree of Figure 9.2, assumption $\lambda\colon p \wedge q$ discharges premise $\mathcal{F} \wedge \mathcal{G}$ of AND-RIGHT-E; derivation tree $\textsc{and-right-e:}\dfrac{\lambda\colon p \wedge q}{q}$ (which has q as conclusion) discharges premise $\mathcal{F}$ of AND-I where $\mathcal{F}$ is instantiated by q; and derivation tree

$$\textsc{and-i:}\dfrac{\textsc{and-right-e:}\dfrac{\lambda\colon\ p \wedge q}{q},\quad \textsc{and-left-e:}\dfrac{\lambda\colon\ p \wedge q}{p}}{q \wedge p}. \tag{9.5}$$

(which has $\lambda\colon p \wedge q$ among its assumptions and has $q \wedge p$ as its conclusion) discharges IMP-I(λ) premise $\dfrac{\overline{\lambda\colon \mathcal{F}}}{\underline{\vdots}}\atop{\mathcal{G}}$ for $\mathcal{F}$ instantiated by $\lambda\colon p \wedge q$ and $\mathcal{G}$ instantiated by $q \wedge p$.

Derivation Trees and Sequents. Derivation trees relate uncanceled assumptions to conclusions in a way that constitutes support for a sequent. We next characterize that relationship. It holds for CFoL of Figure 9.1 as well as for any extension satisfying certain modest requirements, which (by design) CAL satisfies.

> **Soundness for Derivation Trees.** Let L be any logic that has a set of sound inference rules comprising
>
> - IMP-I(λ) and
> - other inference rules whose premises each are given as individual formulas (rather than by more complicated derivation trees).
>
> If a derivation tree $\mathcal{T}$ is constructed using inference rules of logic L then the conclusion of $\mathcal{T}$ will be satisfied provided all uncanceled assumptions of $\mathcal{T}$ are satisfied.[3] □

[3]The result follows by contradiction. Suppose that conclusion $\mathcal{F}$ of some derivation tree $\mathcal{T}$ is *false* but that all uncanceled assumptions $\mathcal{F}_1$, $\mathcal{F}_2$, ..., $\mathcal{F}_n$ of $\mathcal{T}$ are *true*. If we derive a contradiction from this then we prove that $\mathcal{F}$ must be *true* or some $\mathcal{F}_i$ is *false*, which implies what Soundness for Derivation Trees is asserting.

To derive the contradiction we seek, if suffices to observe that if conclusion $\mathcal{F}$ of a derivation tree is *false* then the inference rule whose conclusion was $\mathcal{F}$ must have been instantiated with a premise that is *false*. (This is because, by definition, if all of the premises of any instance of a sound inference rule hold then so will the conclusion. So if the conclusion does not hold then at least one premise must not hold.) By repeatedly invoking the same reasoning backward from

Soundness for Derivation Trees justifies the use of derivation trees as support for sequents. Recall that sequent $\mathcal{F}_1,\ \mathcal{F}_2,\ ...,\ \mathcal{F}_n \vdash_{\mathrm{L}} \mathcal{F}$ when logic L is sound asserts that conclusion $\mathcal{F}$ holds if assumptions $\mathcal{F}_1$, $\mathcal{F}_2$, ..., $\mathcal{F}_n$ hold. Uncanceled assumptions in a derivation tree thus have the same meaning as assumptions in a sequent.

> **Derivation Tree Support for a Sequent.** A derivation tree with conclusion $\mathcal{F}$ and uncanceled assumptions $\mathcal{F}_1$, $\mathcal{F}_2$, ..., $\mathcal{F}_n$ constitutes support for sequent $\mathcal{F}_1,\ \mathcal{F}_2,\ ...,\ \mathcal{F}_n \vdash \mathcal{F}$. □

For example, the derivation tree in Figure 9.2 has no uncanceled assumptions, so it constitutes support for a sequent $\vdash p \wedge q \Rightarrow q \wedge p$; derivation tree (9.5) has two uncanceled assumptions (both are "$p \wedge q$"), so that derivation tree constitutes support for sequent $p \wedge q \vdash q \wedge p$.

The connection between derivation trees and sequents also enables us to use sequents as specifications for derivation trees. Sequent $\mathcal{F}_1,\ \mathcal{F}_2,\ ...,\ \mathcal{F}_n \vdash \mathcal{F}$ specifies a derivation tree with conclusion $\mathcal{F}$ and uncanceled assumptions $\mathcal{F}_1$, $\mathcal{F}_2$, ..., $\mathcal{F}_n$; sequent $\vdash\mathcal{F}$ specifies a derivation tree that has conclusion $\mathcal{F}$ and no uncanceled assumptions. We allow such specifications to appear in isolation or within a derivation tree. So $p \wedge q \vdash q \wedge p$ is a specification for derivation tree (9.5); and in writing

$$\textsc{and-i}\text{:}\ \frac{\lambda{:}p \wedge q \vdash q\ ,\quad \lambda{:}p \wedge q \vdash p}{q \wedge p}$$

we are using sequent $\lambda{:}p \wedge q \vdash q$ to specify a derivation tree having uncanceled assumption $\lambda{:}p \wedge q$ and conclusion q, and we are using sequent $\lambda{:}p \wedge q \vdash p$ to specify an analogous derivation tree but with p as its conclusion.

Inference Rule Details. The inference rules in Figure 9.1 use notation $\mathcal{F}[x := \tau]$ to specify *textual substitution.* This operation distinguishes between *bound occurrences* and *free occurrences* of variables. A bound occurrence of x in a formula $\mathcal{F}$ is an occurrence that appears in the scope of a quantifier (i.e., $\forall$ or $\exists$) where x is named as the *bound variable*; all other occurrences of x in $\mathcal{F}$ are considered free. For instance, in

$$(\forall x:\quad x * y = 0)\ \wedge\ x = 23$$

that premise, we eventually produce a path Π from conclusion $\mathcal{F}$ of the derivation tree upward to some assumption $\lambda{:}\mathcal{F}_i$ of the derivation tree, where $\mathcal{F}_i$ and all other formulas on that path are *false*. Moreover, $\lambda{:}\mathcal{F}_i$ must be a canceled assumption, since we initially supposed that all uncanceled assumptions are *true*. $\mathcal{F}_i$ cannot be a theorem, since $\mathcal{F}_i$ is *false* and theorems are never *false*. So (from the definition of canceled) we conclude that $\mathcal{F}_i$ has a label λ_i and an instance of $\textsc{imp-i}(\lambda_i)$ appears on path Π. According to Figure 9.1, the conclusion of $\textsc{imp-i}(\lambda_i)$ must be "$\mathcal{F}_i \Rightarrow \ldots$". But if $\mathcal{F}_i$ is *false* then "$\mathcal{F}_i \Rightarrow \ldots$". would be *true*, which contradicts the earlier stipulation that all formulas on path Π are *false*. So we derived a contradiction from our supposition that derivation tree conclusion $\mathcal{F}$ is *false* but all uncanceled leaf formulas are *true*.

which is satisfied in those states where $y = 0$ and $x = 23$ are *true*, the first occurrence of x is bound because it appears in the scope of a $\forall x$ quantifier, the occurrence of y is free, and the second occurrence of x is free.

> **Textual Substitution.** $\mathcal{F}[x := \tau]$ is the result of substituting term τ for all free occurrences of variable x in formula $\mathcal{F}$. $\square$

For example, we have:

$$(x > y)[x := z + w] \;=\; (z + w > y) \tag{9.6}$$
$$(\forall x\colon\;\; x > y)[x := z + w] \;=\; (\forall x\colon\;\; x > y) \tag{9.7}$$
$$(\forall x\colon\;\; x > y)[y := z + w] \;=\; (\forall x\colon\;\; x > z + w) \tag{9.8}$$
$$(\forall x\colon\;\; x > y)[y := z + x] \;=\; (\forall x\colon\;\; x > z + x) \tag{9.9}$$

In (9.6), the x in $x > y$ is a free occurrence so $z+w$ is substituted for x, whereas in $(\forall x\colon\;\; x > y)$ the occurrence of x is not free but the occurrence of y is and, therefore, nothing is replaced in (9.7) but $z+w$ does replace y in (9.8). Finally, (9.9) is noteworthy because a free occurrence of x in what is being substituted $(z+x)$ is *captured* by the quantifier and becomes a bound occurrence in resulting formula $(\forall x\colon\;\; x > z + x)$.

The side condition (written in English in Figure 9.1) for FORALL-E and EXISTS-I prevents capture caused by the textual substitutions in those rules. This is needed for soundness, as can be seen by choosing a premise $(\forall x\colon\; \mathcal{F})$ for FORALL-E, where

$$\mathcal{F}\colon\;\; x = 0 \;\;\Rightarrow\;\; (\forall y\colon\;\; y * x = 0).$$

This premise is valid—it asserts that if $x = 0$ holds then so does $x*y = 0$, where y ranges over all possible values. Choosing $y + 1$ for τ in conclusion $\mathcal{F}[x := \tau]$ of FORALL-E violates the side condition of that inference rule because the occurrence of y in $y+1$ is captured when substituted for the x in $(\forall y\colon\;\; y*x = 0)$. If we ignore the side condition, then FORALL-E derives conclusion $\mathcal{F}[x := y+1]$:

$$y + 1 = 0 \;\;\Rightarrow\;\; (\forall y\colon\;\; y * (y + 1) = 0). \tag{9.10}$$

So by ignoring the side condition, we could use FORALL-E to infer (9.10), which is not valid because $y^2 + y = 0$ it does not hold for all values y. The y in antecedent $y + 1 = 0$ of (9.10) refers to the value the state assigns to y but the y in consequent $(\forall y\colon\;\; y * (y + 1) = 0)$ of (9.10) ranges over all possible values. The conclusion of a sound inference rule must always be valid.

Capture causes a formula in which distinct occurrences of the same variable all take the same value to be transformed into a formula where those occurrences may take different values. Assigning a single value to all free occurrences of some variable in a valid formula results in a valid formula; assigning different values to the different occurrences might not result in a valid formula.

The side conditions for FORALL-I and EXISTS-E prevent free occurrences of variables in uncanceled assumptions from being transformed into bound occurrences and *vice versa*. For example, by ignoring the side condition accompanying

FORALL-I, we would obtain derivation tree

$$\text{FORALL-I:}\frac{x = 0}{(\forall x\colon\ x = 0)}. \tag{9.11}$$

We derived a conclusion stipulating that all values equal 0, which is invalid even if uncanceled assumption $x = 0$ of derivation tree (9.11) holds. The free occurrence of x in the uncanceled assumption became a bound occurrence in the conclusion.

The side condition for EXISTS-E prevents a bound occurrence of x in premise $(\exists x\colon \mathcal{F})$ from becoming a free occurrence in conclusion $\mathcal{G}$. Here is a derivation tree that violates this side condition. In it, $\mathcal{F}$ is instantiated by $x = 0$ and $\mathcal{G}$ is instantiated by $x = 0 \vee x = 1$; $(x = 0)[x := 0]$ is deemed to be a canceled assumption because $=$ is reflexive, so "$0 = 0$" is a theorem.

$$\text{EXISTS-E:}\frac{\text{IMP-I}(\lambda)\text{:}\dfrac{\text{OR-LEFT-I:}\dfrac{\boxed{\lambda\colon x = 0}}{x = 0 \vee x = 1}}{x = 0 \Rightarrow (x = 0 \vee x = 1)}\ ,\quad \text{EXISTS-I:}\dfrac{\boxed{(x = 0)[x := 0]}}{(\exists x\colon\ \ x = 0)}}{x = 0 \vee x = 1}$$

There are no uncanceled assumptions in this derivation tree so, according to Soundness for Derivation Trees, the conclusion should be satisfied in all states. Clearly, $x = 0 \vee x = 1$ does not hold in all states—an unsound inference is being made in the derivation tree.

9.3 Extension to Credentials Authorization Logic

Syntax. CAL extends constructive first-order predicate logic CFoL (§9.2) by adding syntax for attribution of beliefs, unrestricted delegation, and restricted delegation. The BNF for CAL formulas $\mathcal{C}$, which uses identifiers P and P' to denote principals, adds the following productions to the BNF given in (9.3) for CFoL formulas $\mathcal{F}$.

$$\begin{aligned}\mathcal{C} ::= \mathcal{F} \ \mid\ P\ \textbf{says}\ \mathcal{C} \ \mid\ P'\ \textbf{speaksfor}\ P \ \mid\ P'\ \textbf{speaks}\ x{:}\mathcal{C}\ \textbf{for}\ P \\ \mid\ \mathcal{C} \wedge \mathcal{C} \ \mid\ \mathcal{C} \vee \mathcal{C} \ \mid\ \mathcal{C} \Rightarrow \mathcal{C}\end{aligned} \tag{9.12}$$

Negation remains an abbreviation:

$$\neg\mathcal{C}\colon\ (\mathcal{C} \Rightarrow \mathit{false}).$$

Notice that CAL formulas can nest attribution and delegation operators, so

$$\texttt{Bob}\ \textbf{says}\ (\texttt{Carol}\ \textbf{says}\ (\texttt{Ted}\ \textbf{speaksfor}\ \texttt{Alice}))$$

is a CAL formula. BNF (9.12) allows quantification to appear only in formulas of CFoL, though. So $(\forall x\colon \texttt{Bob}\ \textbf{says}\ p(x))$ is not a CAL formula, whereas $\texttt{Bob}\ \textbf{says}\ (\forall x\colon p(x))$ is. And quantification over principals is not allowed anywhere in CAL formulas.

Interpretations. The meaning of any logical formula is usually defined relative to some class of *interpretations* by giving a *satisfaction relation* $\models$. If $\iota \models \mathcal{F}$ holds for an interpretation ι and a formula $\mathcal{F}$, then ι is called a *model*[4] for $\mathcal{F}$. Accordingly, ι is defined to be a model for a sequent $\mathcal{F}_1$, $\mathcal{F}_2$, ..., $\mathcal{F}_n \vdash \mathcal{F}$ if ι is a model for conclusion $\mathcal{F}$ whenever ι is a model for each assumption $\mathcal{F}_1$, $\mathcal{F}_2$, ..., $\mathcal{F}_n$. That is, $\iota \models \mathcal{F}_1$, $\iota \models \mathcal{F}_2$, $\ldots$, $\iota \models \mathcal{F}_n$ implies $\iota \models \mathcal{F}$.

A formula $\mathcal{F}$ is *valid* (denoted $\models \mathcal{F}$) if and only if every interpretation is a model for $\mathcal{F}$; a sequent $\mathcal{F}_1$, $\mathcal{F}_2$, ..., $\mathcal{F}_n \vdash \mathcal{F}$ is valid if and only if every interpretation ι is a model for that sequent. This definition of a valid sequent is consistent with the connection between sequents and derivation trees discussed in Derivation Tree Support for a Sequent (page 210) in light of Soundness of Derivation Trees (page 209)

Interpretations for a constructive logic are intended to capture the knowledge that might be available to some reasoning agent, such as a guard. Additional information leads to increased levels of knowledge. An *accessibility relation* $\succeq$ can be used to characterize increased levels of knowledge: $\iota' \succeq \iota$ holds if interpretation ι' contains all knowledge represented by interpretation ι (and ι' may contain additional knowledge, too).

The satisfaction relation $\models$ for a constructive logic is required to be monotonic with respect to increased levels of knowledge:

Satisfaction Monotonicity. If $\iota \models \mathcal{F}$ and $\iota' \succeq \iota$ then $\iota' \models \mathcal{F}$. □

That is, adding knowledge to an interpretation ι that is a model for some formula $\mathcal{F}$, always produces an interpretation ι' that is also a model for $\mathcal{F}$. Satisfaction Monotonicity makes constructive logics well suited for use by guards. A guard typically has incomplete information about the system state. Suppose, based on that information, the guard establishes that some goal formula $\mathcal{F}$ is satisfied, and thus an access should be allowed to proceed. Satisfaction Monotonicity implies that having additional information about that system state does not lead to a different decision—$\mathcal{F}$ is satisfied in the resulting interpretation, too.

Formulas of ordinary first-order predicate logics typically have states as interpretations. Each state σ associates a value with each variable. Interpretation σ is a model for a formula $\mathcal{F}$ if *true* is the result of evaluating the expression obtained when free occurrences of variables in $\mathcal{F}$ are replaced by values σ assigns to those variables. For instance, if σ associates value 23 with x then we would have $\sigma \models x > 0$ and say that σ is a model for $x > 0$.

A richer class of interpretations must be employed for a constructive first-order predicate logic. Here, a *partial state* associates values with some variables but need not associate a value with every variable, and $\sigma' \succeq \sigma$ holds if every variable assigned a value by partial state σ is assigned the same value by partial state σ' (but σ' may assign values to additional variables, too). Satisfaction Monotonicity then restricts satisfaction relation $\models_{\text{CFoL}}$ for CFoL so that $\sigma \models_{\text{CFoL}} \mathcal{F}$ implies $\sigma' \models_{\text{CFoL}} \mathcal{F}$ for $\sigma' \succeq \sigma$: if some partial state σ is a model for

[4]For this reason "$\iota \models \mathcal{F}$" is often read as "$\iota$ models $\mathcal{F}$"

$\mathcal{F}$ and additional information takes the reasoning agent to some partial state $\sigma' \succeq \sigma$ then σ' also will be a model for $\mathcal{F}$.

States and partial states do not include information about beliefs held by principals. CAL is concerned with reasoning about such beliefs, so states and partial states alone cannot serve as interpretations for CAL formulas. A CAL interpretation ι instead is a pair $\langle \sigma_\iota,\ \omega_\iota(\cdot)\rangle$, where

- σ_ι is a partial state, and
- *worldview* $\omega_\iota(\cdot)$ is a mapping from principals to sets of CAL formulas.

Each CAL formula in $\omega_\iota(P)$ conveys a belief that principal P holds. For example, we would have

$$\sigma_\iota \models_{\mathrm{CFoL}} x = 0 \qquad \text{“}x > 0\text{”} \in \omega_\iota(P_1) \qquad \text{“}x \le 0\text{”} \in \omega_\iota(P_2) \qquad (9.13)$$

for a CAL interpretation ι in which partial state σ_ι assigns value 0 to x, principal P_1 holds belief $x > 0$, and principal P_2 holds belief $x \le 0$. So ι describes a situation in which P_1 is misinformed about the state and P_2 has incomplete information.

CAL is a constructive logic, which requires that an accessibility relation $\succeq$ be defined on CAL interpretations. We extend accessibility relation $\succeq$ on partial states to CAL interpretations by equating expanded sets of beliefs with higher levels of knowledge. So, for CAL interpretations ι and ι', we define $\iota' \succeq \iota$ to hold if and only if

$$\sigma_{\iota'} \succeq \sigma_\iota \qquad \wedge \qquad \omega_{\iota'}(P) \supseteq \omega_\iota(P) \text{ for all principals } P. \qquad (9.14)$$

Satisfaction relation $\iota \models_{\mathrm{CAL}} \mathcal{C}$ for a CAL interpretation ι and a CAL formula $\mathcal{C}$ can now be given.

Formal Definition of CAL Satisfaction Relation $\models_{\mathrm{CAL}}$. The formal definition of

$$\iota \models_{\mathrm{CAL}} \mathcal{C}$$

proceeds by cases, according to the the syntax of $\mathcal{C}$. For each case, we also show that Satisfaction Monotonicity is obeyed, so that we can conclude (by structural induction) that Satisfaction Monotonicity is obeyed for all CAL formulas.

$\mathcal{F}$ a formula of Constructive First-Order Predicate Logic CFoL is satisfied in all those interpretations $\iota = \langle \sigma_\iota, \omega_\iota(\cdot)\rangle$ where state σ_ι is a model for $\mathcal{F}$:

$$\iota \models_{\mathrm{CAL}} \mathcal{F} \quad \text{iff} \quad \sigma_\iota \models_{\mathrm{CFoL}} \mathcal{F}$$

Satisfaction Monotonicity follows because $\models_{\mathrm{CFoL}}$ obeys Satisfaction Monotonicity.

P **says** $\mathcal{C}$ is satisfied in all those interpretations $\iota = \langle \sigma_\iota, \omega_\iota(\cdot)\rangle$ where principal P holds belief $\mathcal{C}$:

$$\iota \models_{\text{CAL}} P\ \textbf{says}\ \mathcal{C} \quad \text{iff} \quad \mathcal{C} \in \omega_\iota(P)$$

Satisfaction Monotonicity follows because definition (9.14) implies that $\omega_{\iota'}(P) \supseteq \omega_\iota(P)$ holds whenever $\iota' \succeq \iota$ does. Therefore, $\mathcal{C} \in \omega_\iota(P)$ implies $\mathcal{C} \in \omega_{\iota'}(P)$, and we conclude that $\iota' \models_{\text{CAL}} P$ **says** $\mathcal{C}$ holds when $\iota' \models_{\text{CAL}} P$ **says** $\mathcal{C}$ does.

P **speaksfor** P' is satisfied in an interpretation $\iota = \langle \sigma_\iota, \omega_\iota(\cdot)\rangle$ provided that for ι or any interpretation $\iota' = \langle \sigma_{\iota'}, \omega_{\iota'}(\cdot)\rangle$ corresponding to a higher level of knowledge, all beliefs that P holds are beliefs that P' holds too:

$$\iota \models_{\text{CAL}} P\ \textbf{speaksfor}\ P' \quad \text{iff} \quad \omega_{\iota'}(P) \subseteq \omega_{\iota'}(P') \text{ for all } \iota' \succeq \iota$$

Satisfaction Monotonicity follows directly from the quantification over ι'. Had we omitted that quantification and instead simply required $\omega_\iota(P) \subseteq \omega_\iota(P')$ then Satisfaction Monotonicity would not be guaranteed. An example is $\iota' \succeq \iota$ where $\omega_\iota(P) \subseteq \omega_\iota(P')$ but $\omega_{\iota'}(P) \not\subseteq \omega_{\iota'}(P')$, because $\omega_\iota(P') = \omega_{\iota'}(P')$ and $\omega_\iota(P) \subset \omega_{\iota'}(P)$ hold.

P **speaks** $x{:}\mathcal{C}$ **for** P' is satisfied in an interpretation $\iota = \langle \sigma_\iota, \omega_\iota(\cdot)\rangle$ provided that in ι or any interpretation $\iota' = \langle \sigma_{\iota'}, \omega_{\iota'}(\cdot)\rangle$ corresponding to a higher level of knowledge, all beliefs of the form $\mathcal{C}[x := \tau]$ that P holds (for any term τ) are also beliefs that P' holds. Formally:

$$\iota \models_{\text{CAL}} P\ \textbf{speaks}\ x{:}\mathcal{C}\ \textbf{for}\ P' \quad \text{iff} \quad \omega_{\iota'}(P)|_{x:\mathcal{C}} \subseteq \omega_{\iota'}(P') \text{ for all } \iota' \succeq \iota$$

where $\omega_\iota(P)|_{x:\mathcal{C}}$ is defined to be the largest subset of $\omega_\iota(P)$ in which all beliefs have form $\mathcal{C}[x := \tau]$ for any term τ. The argument that Satisfaction Monotonicity is obeyed here is analogous to the one given above for P **speaksfor** P'.

Conjunctions and Disjunctions are satisfied in an interpretation $\iota = \langle \sigma_\iota, \omega_\iota(\cdot)\rangle$ according to the usual meanings given to propositional connectives $\wedge$ and $\vee$:

$$\begin{array}{lll} \iota \models_{\text{CAL}} \mathcal{C} \wedge \mathcal{C}' & \text{iff} & \iota \models_{\text{CAL}} \mathcal{C} \text{ and } \iota \models_{\text{CAL}} \mathcal{C}' \\ \iota \models_{\text{CAL}} \mathcal{C} \vee \mathcal{C}' & \text{iff} & \iota \models_{\text{CAL}} \mathcal{C} \text{ or } \iota \models_{\text{CAL}} \mathcal{C}' \end{array}$$

Satisfaction Monotonicity follows by structural induction. We sketch the argument for a conjunction $\mathcal{C} \wedge \mathcal{C}'$. (Disjunction is similar.) Assume $\iota \models_{\text{CAL}} \mathcal{C} \wedge \mathcal{C}'$ so, by definition $\iota \models_{\text{CAL}} \mathcal{C}$ and $\iota \models_{\text{CAL}} \mathcal{C}'$. Satisfaction Monotonicity for $\iota \models_{\text{CAL}} \mathcal{C}$ and for $\iota \models_{\text{CAL}} \mathcal{C}'$ then implies $\iota' \models_{\text{CAL}} \mathcal{C}$ and $\iota' \models_{\text{CAL}} \mathcal{C}'$ for $\iota' \succeq \iota$, which (by definition) implies $\iota' \models_{\text{CAL}} \mathcal{C} \wedge \mathcal{C}'$. And $\iota' \models_{\text{CAL}} \mathcal{C} \wedge \mathcal{C}'$ for $\iota' \succeq \iota$ is what Satisfaction Monotonicity requires.

Implication $\mathcal{C} \Rightarrow \mathcal{C}'$ is satisfied in an interpretation $\iota = \langle \sigma_\iota, \omega_\iota(\cdot) \rangle$ according to the usual meaning of propositional connective $\Rightarrow$ in all interpretations $\iota' \succeq \iota$:

$$\iota \models_{\mathrm{CAL}} \mathcal{C} \Rightarrow \mathcal{C}' \quad \text{iff} \quad \iota' \models_{\mathrm{CAL}} \mathcal{C} \text{ implies } \iota' \models_{\mathrm{CAL}} \mathcal{C}' \text{ for all } \iota' \succeq \iota$$

As with **speaksfor**, Satisfaction Monotonicity requires the quantification over ι'. The quantification ensures $\iota \not\models_{\mathrm{CAL}} \mathcal{C} \Rightarrow \mathcal{C}'$ for the case where ι is a model for neither $\mathcal{C}$ or $\mathcal{C}'$ (so $\iota \models_{\mathrm{CAL}} \mathcal{C}$ implies $\iota \models_{\mathrm{CAL}} \mathcal{C}'$) but CAL interpretation $\iota' \succeq \iota$ is a model for $\mathcal{C}$ but not a model for $\mathcal{C}'$ (so $\iota' \models_{\mathrm{CAL}} \mathcal{C}$ implies $\iota' \models_{\mathrm{CAL}} \mathcal{C}'$ does not hold). □

Worldview Anatomy. Worldview $\omega(P)$, which contains the set[5] of beliefs held by a principal P, constitutes the basis for credentials P issues. We formalize this connection between beliefs in $\omega(P)$ and credentials issued by P as:

Credentials Foundation. A principal P issues a credential conveying P **says** $\mathcal{C}$—thereby attesting to a belief $\mathcal{C}$ that P holds—only if $\mathcal{C} \in \omega(P)$ is *true*. □

When P is operating correctly, beliefs in $\omega(P)$ derive from (i) credentials P receives, (ii) other inputs, (iii) system state P reads, and (iv) the internal logic of programs P executes (because programs embody beliefs through what they compute). Notice that if P is compromised and issues bogus credentials, then Credentials Foundation implies that $\omega(P)$ contains the corresponding bogus beliefs—even if it means $\omega(P)$ contains mutually inconsistent beliefs.

Different programs that a principal P might execute will contribute different beliefs to worldview $\omega(P)$. Rather than analyzing each specific program for its contributions to $\omega(P)$, we instead employ a conservative approximation for worldviews and, henceforth, restrict consideration to only those interpretations that contain such a conservative approximation. Specifically, we posit that worldview $\omega(P)$ contains (i) a set Init_P of beliefs that reflect those aspects of the initial system state known to P and (ii) all logical consequences. Presumably, beliefs contributed by programs P executes would be some subset of those logical consequences, which is what makes our worldviews conservative approximations.

The procedure for constructing a conservative approximation enumerates the (potentially infinite) set of logical consequences, starting with set Init_P of beliefs.

Conservative Approximation for Worldviews. For a principal P having a set Init_P of beliefs, worldview $\omega(P)$ is defined by

$$\omega(P) = \mathit{cl}_{\mathrm{CAL}}(P, \mathit{Init}_P)$$

where *deductive closure* $\mathit{cl}_{\mathrm{CAL}}(P, B)$ is computed by starting with set B and repeatedly adding CAL formulas:

[5]For a system whose current state is described by CAL interpretation $\iota = \langle \sigma_\iota, \omega_\iota(\cdot) \rangle$, we would thus have $\omega(P) = \omega_\iota(P)$.

(i) Add all valid formulas of CAL.

(ii) Add formulas $\mathcal{C}$ for which

$$P\ \textbf{says}\ \mathcal{C}_1,\ P\ \textbf{says}\ \mathcal{C}_2,\ \ldots,\ P\ \textbf{says}\ \mathcal{C}_N \vdash_{\mathrm{CAL}} P\ \textbf{says}\ \mathcal{C}$$

if $\mathcal{C}_i \in cl_{\mathrm{CAL}}(P, B)$ holds for $1 \leq i \leq N$.

(iii) Add all formulas in $\omega(P')$ if "P' **speaksfor** P" $\in cl_{\mathrm{CAL}}(P, B)$.

(iv) Add all formulas $\mathcal{C}[x := \tau]$ for any term τ where $\mathcal{C}[x := \tau] \in \omega(P')$ and "P' **speaks** $x{:}\mathcal{C}$ **for** P" $\in cl_{\mathrm{CAL}}(P, B)$. □

Clause (i) adds to $\omega(P)$ all CAL theorems. Clauses (ii) – (iv) incorporate into $\omega(P)$ any beliefs that are logical consequences of other beliefs in $\omega(P)$, including logical consequences of beliefs previously added by clauses (i) – (iv).

Principals may hold inconsistent beliefs. A principal might receive inconsistent credentials, obtain inconsistent values from reading a changing system state at different instants, or execute programs that are buggy or malicious. Credentials issued by a principal P holding inconsistent beliefs $\mathcal{B}$ and $\neg\mathcal{B}$ (say) are problematic for making authorization decisions. Here's why. $\neg\mathcal{B}$ is an abbreviation for $\mathcal{B} \Rightarrow \mathit{false}$, so IMP-E (Figure 9.3) with premises B and $\neg\mathcal{B}$ would add *false* to $\omega(P)$, due to clause (ii) of Conservative Approximation for Worldviews. Applications of inference rule FALSE (Figure 9.3) then adds to $\omega(P)$ any and all CAL formulas. So when P holds inconsistent beliefs, credentials from P should not be used to justify authorizing a request. Fortunately, inconsistency in credentials issued by a principal P often can be detected; guards can then be notified that credentials with source P should be ignored.

CAL Inference Rules. The inference rules of CAL are sound for interpretations constructed using our conservative approximations of worldviews. The set of inference rules comprise those of CFoL (Figure 9.1), their counterparts (Figure 9.3) for reasoning about CAL formulas constructed using propositional connectives ($\wedge$, $\vee$, and $\Rightarrow$), and inference rules (Figure 9.4) for reasoning about **says**, unrestricted delegation **speaksfor**, and restricted delegation **speaks** $x{:}\mathcal{C}$ **for**. This last set of inference rules merits further discussion.

SAYS-I asserts with its conclusion P **says** $\mathcal{C}$ that worldview $\omega(P)$ of each principal P contains every previously proved theorem $\mathcal{C}$; clause (i) of Conservative Approximations for Worldviews allows this. The restriction (conveyed by writing $\vdash_{\mathrm{CAL}} \mathcal{C}$ as the premise) that a SAYS-I premise must be a derivation tree with no uncanceled assumptions warrants explanation. If the premise of SAYS-I could instead be discharged by exhibiting a derivation tree with uncanceled assumptions then SAYS-I, in conjunction with IMP-I(λ), could be used to derive support[6] for sequent $\vdash_{\mathrm{CAL}} \mathcal{C} \Rightarrow P$ **says** $\mathcal{C}$. That sequent is sound

[6]Here is a derivation tree for sequent $\vdash_{\mathrm{CAL}} \mathcal{C} \Rightarrow P$ **says** $\mathcal{C}$ assuming the premise for SAYS-I could be discharged using an arbitrary CAL formula $\mathcal{C}$.

$$\textsc{imp-i}(\lambda)\text{:}\ \dfrac{\textsc{says-i}\text{:}\ \dfrac{\boxed{\lambda{:}\mathcal{C}}}{P\ \textbf{says}\ \mathcal{C}}}{\mathcal{C} \Rightarrow P\ \textbf{says}\ \mathcal{C}}$$

$$\text{AND-I: }\frac{\mathcal{C}\ ,\ \mathcal{C}'}{\mathcal{C}\wedge\mathcal{C}'}\qquad \text{AND-LEFT-E: }\frac{\mathcal{C}\wedge\mathcal{C}'}{\mathcal{C}}\qquad \text{AND-RIGHT-E: }\frac{\mathcal{C}\wedge\mathcal{C}'}{\mathcal{C}'}$$

$$\text{OR-LEFT-I: }\frac{\mathcal{C}}{\mathcal{C}\vee\mathcal{C}'}\qquad \text{OR-RIGHT-I: }\frac{\mathcal{C}'}{\mathcal{C}\vee\mathcal{C}'}\qquad \text{OR-E: }\frac{\mathcal{C}\Rightarrow\mathcal{C}''\ ,\ \mathcal{C}'\Rightarrow\mathcal{C}''\ ,\ \mathcal{C}\vee\mathcal{C}'}{\mathcal{C}''}$$

$$\text{IMP-E: }\frac{\mathcal{C}\ ,\ \mathcal{C}\Rightarrow\mathcal{C}'}{\mathcal{C}'}\qquad \text{IMP-I}(\lambda)\text{: }\frac{\begin{array}{c}\underline{\lambda\text{: }\mathcal{C}}\\ \vdots\\ \overline{\mathcal{C}'}\end{array}}{\mathcal{C}\Rightarrow\mathcal{C}'}\qquad \text{FALSE: }\frac{\mathit{false}}{\mathcal{C}}$$

Figure 9.3: CAL Inference Rules for Propositional Connectives

only if every principal P holds a belief $\mathcal{C}$ whenever $\mathcal{C}$ is *true*—an omniscience assumption about principals that is unrealistic, because a principal might well be ignorant about some aspects of the system state or about beliefs other principals hold. For instance, were we requiring that $\vdash_{\mathrm{CAL}} \mathcal{C} \Rightarrow P$ **says** $\mathcal{C}$ be sound then P **says** $(x = 0)$ would have to be *true* if $x = 0$ is *true*, even when variable x is located on some distant computer that is not communicating with P.

SAYS^2-I and SAYS-E concern introspection. SAYS^2-I asserts that if $\mathcal{C} \in \omega(P)$ is *true*, so P **says** $\mathcal{C}$ holds, then P is sufficiently introspective to have that "P **says** $\mathcal{C}$" $\in \omega(P)$ is *true* too, so P **says** $(P\ \textbf{says}\ \mathcal{C})$ holds. SAYS-E then ensures that introspective beliefs have a basis: if "P **says** $\mathcal{C}$" $\in \omega(P)$ it *true* then so is "$\mathcal{C}$" $\in \omega(P)$. SAYS-E would be superfluous if SAYS-I and SAYS^2-I were the only ways to derive P **says** $(P\ \textbf{says}\ \mathcal{C})$. However, other CAL inference rules (e.g., SAYS-IMP-E, DELEG-E, and REST-DELEG-E) also can derive conclusions containing P **says** $(P\ \textbf{says}\ \mathcal{C})$; SAYS-E allows P **says** $\mathcal{C}$ to be deduced no matter how P **says** $(P\ \textbf{says}\ \mathcal{C})$ has been derived.

A typical use of SAYS-IMP-E is illustrated by the derivation tree for conclusion P **says** $\mathcal{C}'$ from assumptions P **says** $\mathcal{C}$ and P **says** $(\mathcal{C} \Rightarrow \mathcal{C}')$.

$$\text{IMP-E: }\frac{P\ \textbf{says}\ \mathcal{C},\quad \text{SAYS-IMP-E: }\dfrac{P\ \textbf{says}\ (\mathcal{C}\Rightarrow\mathcal{C}')}{(P\ \textbf{says}\ \mathcal{C})\Rightarrow(P\ \textbf{says}\ \mathcal{C}')}}{P\ \textbf{says}\ \mathcal{C}'} \tag{9.15}$$

Notice that a single principal (*viz.* P) serves as the source in the premises and for the conclusion of IMP-E, so inconsistency cannot be produced at one principal by using IMP-E to combine beliefs that other principals hold.

This raises a broader question: Can a combination of CAL rules be used to derive an inconsistency in the worldview at one principal by combining beliefs that other principals hold? The answer is no. We define two or more principals

$$\textsc{says-i}: \frac{\vdash_{\text{CAL}} \mathcal{C}}{P \textbf{ says } \mathcal{C}} \qquad \textsc{says}^2\textsc{-i}: \frac{P \textbf{ says } \mathcal{C}}{P \textbf{ says } (P \textbf{ says } \mathcal{C})} \qquad \textsc{says-e}: \frac{P \textbf{ says } (P \textbf{ says } \mathcal{C})}{P \textbf{ says } \mathcal{C}}$$

$$\textsc{says-imp-e}: \frac{P \textbf{ says } (\mathcal{C} \Rightarrow \mathcal{C}')}{(P \textbf{ says } \mathcal{C}) \Rightarrow (P \textbf{ says } \mathcal{C}')}$$

(a) Inference Rules for **says**

$$\textsc{hand-off}: \frac{P \textbf{ says } (P' \textbf{ speaksfor } P)}{P' \textbf{ speaksfor } P} \qquad \textsc{deleg-e}: \frac{P' \textbf{ speaksfor } P}{(P' \textbf{ says } \mathcal{C}) \Rightarrow (P \textbf{ says } \mathcal{C})}$$

$$\textsc{deleg-trans}: \frac{P \textbf{ speaksfor } P',\ P' \textbf{ speaksfor } P''}{P \textbf{ speaksfor } P''}$$

(b) Inference Rules for Unrestricted Delegation (**speaksfor**)

$$\textsc{rest-narrow}: \frac{P' \textbf{ speaksfor } P}{P' \textbf{ speaks } x{:}\mathcal{C} \textbf{ for } P}$$

$$\textsc{rest-hand-off}: \frac{P \textbf{ says } (P' \textbf{ speaks } x{:}\mathcal{C} \textbf{ for } P)}{P' \textbf{ speaks } x{:}\mathcal{C} \textbf{ for } P}$$

$$\textsc{rest-deleg-e}: \frac{P' \textbf{ speaks } x{:}\mathcal{C} \textbf{ for } P}{(P' \textbf{ says } \mathcal{C}[x := \tau]) \quad \Rightarrow \quad (P \textbf{ says } \mathcal{C}[x := \tau])}$$

$$\textsc{rest-deleg-trans}: \frac{P \textbf{ speaks } x{:}\mathcal{C} \textbf{ for } P',\ P' \textbf{ speaks } x{:}\mathcal{C} \textbf{ for } P''}{P \textbf{ speaks } x{:}\mathcal{C} \textbf{ for } P''}$$

(c) Inference Rules for Restricted Delegation (**speaks** $x{:}\mathcal{C}$ **for**)

Figure 9.4: CAL Inference Rules for **says** and **speaksfor**

to be *independent* if none either directly or indirectly makes an unrestricted or restricted delegation to another.

> **Non-interference in CAL.** Let $\mathit{IP} = \{P_1, P_2, ..., P_n\}$ be any set of independent principals. For every $P \in \mathit{IP}$
>
> $$\mathcal{C}_1, \ldots, \mathcal{C}_m \vdash_{\text{CAL}} P \textbf{ says } \mathit{false}$$
>
> if and only if
>
> $$\mathcal{C}'_1, \ldots, \mathcal{C}'_p \vdash_{\text{CAL}} P \textbf{ says } \mathit{false}$$
>
> where no assumption $\mathcal{C}'_i$ includes "P_j **says** ..." for $P_j \in \mathit{IP} - \{P\}$. □

Non-interference in CAL implies that inconsistency in the worldview of one principal cannot be the result of beliefs that other, independent, principals hold. For example, if P and P' are independent principals then P **says** *false* cannot contribute to a derivation of P' **says** *false*.

HAND-OFF asserts that a principal P is the one to decide whether to adopt the set of beliefs held by some other principal P'. Moreover, by holding belief P' **speaksfor** P, a principal P becomes accountable by HAND-OFF for all beliefs held by principal P' and, in so doing, delegates its full authority to P'. Soundness of HAND-OFF follows from clause (iii) of Conservative Approximation for Worldviews, as follows. If premise P **says** (P' **speaksfor** P) is satisfied then we conclude "P' **speaksfor** P" $\in \omega(P)$ is *true*, so clause (iii) implies that $\mathcal{C}$ is added to $\omega(P)$ for all $\mathcal{C} \in \omega(P')$. Thus, by construction, $\omega(P') \subseteq \omega(P)$, which means conclusion P' **speaksfor** P of HAND-OFF is satisfied, and HAND-OFF is sound.

The consequences of unrestricted delegation are materialized with DELEG-E. Here is a derivation tree to conclude P **says** $\mathcal{C}$ from assumptions P' **says** $\mathcal{C}$ and P' **speaksfor** P.

$$\text{IMP-E: }\frac{P' \textbf{ says } \mathcal{C}, \quad \text{DELEG-E: }\dfrac{P' \textbf{ speaksfor } P}{(P' \textbf{ says } \mathcal{C}) \Rightarrow (P \textbf{ says } \mathcal{C})}}{P \textbf{ says } \mathcal{C}} \tag{9.16}$$

To be convinced that DELEG-E is sound, observe that if premise P' **speaksfor** P is satisfied then by definition $\omega(P') \subseteq \omega(P)$, so $\mathcal{C} \in \omega(P') \Rightarrow \mathcal{C} \in \omega(P)$ holds. Consequently (P' **says** $\mathcal{C}$) $\Rightarrow$ (P **says** $\mathcal{C}$), the conclusion of DELEG-E, is satisfied.

DELEG-TRANS surfaces inferences from the transitivity of unrestricted delegation. If P' **speaksfor** P'' holds then not only are all beliefs in $\omega(P')$ incorporated into $\omega(P'')$ but so are all beliefs in $\omega(P)$ for principals P that P' delegates to (hence trusts) directly or transitively. To trust a principal P' thus not only means adopting the beliefs that P' holds but also trusting choices P' makes about which principals it trusts, principals they trust, and so on. The risk of inconsistency in a worldview increases by incorporating beliefs from sets of known and unknown principals, so transitivity of delegation can bring unpleasant surprises. Transitivity of delegation is useful, however, for allowing clients to be ignorant about implementation details for services they invoke.

Only when delegation is transitive can a service be implemented in terms of other services without also requiring that its clients both know the identities of the other services and make explicit delegations to those other services.

REST-NARROW, REST-HAND-OFF, REST-DELEG-E, and REST-DELEG-TRANS provide a means to mitigate some of the risks that unrestricted delegation brings. By using REST-HAND-OFF, a source becomes accountable for beliefs that have some pre-specified form $\mathcal{C}[x := \tau]$ for a variable x and term τ. For example,

$$\texttt{CSdept}\ \textbf{says}\ (\texttt{Univ}\ \textbf{speaks}\ x{:}\mathit{Enrolled}(x)\ \textbf{for}\ \texttt{CSdept}) \tag{9.17}$$

is satisfied when $\omega(\texttt{CSdept})$ incorporates the subset of beliefs `Univ` holds and have form "$\mathit{Enrolled}(x)$", where x has been replaced by some value. We might have

$$\texttt{Univ}\ \textbf{says}\ \mathit{Enrolled}(\texttt{MMB})$$
$$\texttt{Univ}\ \textbf{says}\ \neg\mathit{Enrolled}(\texttt{MMB})$$

which would mean that `Univ` holds inconsistent beliefs. Under unrestricted delegation

$$\texttt{Cornell}\ \textbf{speaksfor}\ \texttt{CSdept} \tag{9.18}$$

`CSdept` incorporates beliefs $\mathit{Enrolled}(\texttt{MMB})$ and $\neg\mathit{Enrolled}(\texttt{MMB})$, which makes $\omega(\texttt{CSdept})$ inconsistent. Replace unrestricted delegation (9.18) by the restricted delegation

$$\texttt{Univ}\ \textbf{speaks}\ x{:}\mathit{Enrolled}(x)\ \textbf{for}\ \texttt{CSdept} \tag{9.19}$$

and this inconsistency is eliminated from $\omega(\texttt{CSdept})$, because (9.19) adds belief $\mathit{Enrolled}(\texttt{MMB})$ but not $\neg\mathit{Enrolled}(\texttt{MMB})$ into $\omega(\texttt{CSdept})$. Of course, (9.19) could still lead to inconsistency in $\omega(\texttt{CSdept})$ if other inferences lead `CSdept` to hold belief $\neg\mathit{Enrolled}(\texttt{MMB})$.

REST-DELEG-E not only concerns beliefs represented by a single formula $\mathcal{C}$ but, in combination with SAYS-IMP-E, also applies to beliefs implied by $\mathcal{C}$. For example, P **says** $\mathcal{C}'$ can be derived if (i) P' holds a belief $\mathcal{C}$ named in a restricted delegation from P to P' and (ii) $\mathcal{C} \Rightarrow \mathcal{C}'$ is a CAL theorem:

$$\textsc{imp-e}{:}\ \dfrac{\textsc{imp-e}{:}\ \dfrac{P'\ \textbf{says}\ \mathcal{C},\ \ \textsc{deleg-e}{:}\ \dfrac{P'\ \textbf{speaks}\ x{:}\mathcal{C}\ \textbf{for}\ P}{(P'\ \textbf{says}\ \mathcal{C}) \Rightarrow (P\ \textbf{says}\ \mathcal{C})}}{P\ \textbf{says}\ \mathcal{C}} \qquad \textsc{says-imp-e}{:}\ \dfrac{\textsc{says-i}{:}\ \dfrac{\vdash_{\text{CAL}} \mathcal{C} \Rightarrow \mathcal{C}'}{P\ \textbf{says}\ (\mathcal{C} \Rightarrow \mathcal{C}')}}{(P\ \textbf{says}\ \mathcal{C}) \Rightarrow (P\ \textbf{says}\ \mathcal{C}')}}{P\ \textbf{says}\ \mathcal{C}'}$$

Some helpful derived inference rules of CAL are given in Figure 9.5. They facilitate proofs that are shorter and/or easier to construct than proofs that use only the inference rules found in Figure 9.1, Figure 9.3, and Figure 9.4. Any proof that uses a derived inference rule can, by definition, be mechanically transformed into a proof that does not use that derived inference rule. For example, each instance of SAYS-IMP-MP can be replaced by a version of derivation tree (9.15).

$$\text{SAYS-AND-I: }\frac{(P \textbf{ says } \mathcal{C}) \wedge (P \textbf{ says } \mathcal{C}')}{P \textbf{ says } (\mathcal{C} \wedge \mathcal{C}')} \qquad \text{SAYS-AND-E: }\frac{P \textbf{ says } (\mathcal{C} \wedge \mathcal{C}')}{(P \textbf{ says } \mathcal{C}) \wedge (P \textbf{ says } \mathcal{C}')}$$

$$\text{SAYS-OR-I: }\frac{(P \textbf{ says } \mathcal{C}) \vee (P \textbf{ says } \mathcal{C}')}{P \textbf{ says } (\mathcal{C} \vee \mathcal{C}')} \qquad \text{SAYS-OR-E: }\frac{P \textbf{ says } (\mathcal{C} \vee \mathcal{C}')}{(P \textbf{ says } \mathcal{C}) \vee (P \textbf{ says } \mathcal{C}')}$$

$$\text{SAYS-IMP-I: }\frac{(P \textbf{ says } \mathcal{C}) \Rightarrow (P \textbf{ says } \mathcal{C}')}{P \textbf{ says } (\mathcal{C} \Rightarrow \mathcal{C}')} \qquad \text{SAYS-IMP-MP: }\frac{P \textbf{ says } \mathcal{C},\ P \textbf{ says } (\mathcal{C} \Rightarrow \mathcal{C}')}{P \textbf{ says } \mathcal{C}'}$$

Figure 9.5: Useful Derived Inference Rules for CAL

9.4 Compound Principals

Any system component P—whether it is implemented by hardware, software, or some combination—can be considered a CAL principal provided it can be assigned a worldview $\omega(P)$ as defined in Conservative Approximation for Worldviews (page 216). When P comprises multiple components that are themselves CAL principals, $\omega(P)$ incorporates beliefs from the worldviews of those components. CAL **speaksfor** operator enables explicit declarations that the worldview of some principal includes the worldview of another. In this section, we discuss how a syntax for principal names can offer an implicit means to convey relationships between worldviews.

Subprincipals. For any principal P and any *qualifier* η that is a term ranging over a set of values (including distinguished value ε), $P.\eta$ denotes a *subprincipal* of P. Thus, among the subprincipals of P are: $P.5$, $P.id$, and $P.(hour \bmod 24)$. $P.\eta$ is defined to be the principal having worldview

$$\omega(P.\eta) \;=\; cl_{\text{CAL}}(P.\eta,\ Init_{P.\eta} \cup \{P \textbf{ speaksfor } P.\eta\}). \qquad (9.20)$$

Based on this definition, Conservative Approximation for Worldviews clause (iii) implies that $\omega(P) \subseteq \omega(P.\eta)$ holds, and therefore the following CAL inference rule is sound.

$$\text{SUBPRIN: }\frac{}{P \textbf{ speaksfor } P.\eta}$$

The qualifier η used for naming subprincipal $P.\eta$ provides a basis to distinguish among subprincipals of a given principal.

$$\text{EQUIV-SUBPRIN: }\frac{\eta = \eta'}{P.\eta \textbf{ speaksfor } P.\eta'}$$

Because subprincipal $P.\eta$ is itself a principal, it can have subprincipals; we assume left-associativity, so $P.\eta.\eta'$ abbreviates $(P.\eta).\eta'$.

Notice, SUBPRIN allows any statement by a principal P to be attributed to any subprincipal of P. That is, from P **says** $\mathcal{C}$ we can derive $P.\eta$ **says** $\mathcal{C}$ for any subprincipal $P.\eta$ of P. Unintended attributions are avoided by adopting a naming convention. We might, for example, agree to attribute to subprincipal $P.\varepsilon$ any belief by P that should not be attributed to any other subprincipal $P.\eta$ of P. $P.\eta$ is not a subprincipal of $P.\varepsilon$, so beliefs attributed to $P.\varepsilon$ are not inherited by $P.\eta$.

One common use of subprincipals is for defining different instances of a principal, where each instance authorizes requests issued during disjoint epochs or associated with different nonces. A single component *FileSys* might be realized using a set *FileSys*.1, *FileSys*.2, ..., *FileSys*.i, ... of subprincipals that are each responsible for handling disjoint subsets of requests to access data stored in a single shared file system. By including in the goal formula for subprincipal *FileSys*.i the conjunct "$i = current$" (where *current* is an integer variable accessible to all of the subprincipals), only the "current" instance *FileSys*.*current* of *FileSys* ever authorizes requests.

Subprincipals are also useful when one principal is implemented in terms of another. A process is implemented by multiplexing a hardware processor; script execution is implemented by an interpreter; and a communications channels could be implemented by multiplexing a wire or fiber. In general, one component L *implements* another component H if all actions being attributed to H are actually performed by L. Because actions a principal performs are based on beliefs in its worldview, a CAL characterization for whether a principal P_L implements principal P_H would stipulate that P_H **says** $\mathcal{C}$ holds only if it can be derived from P_L **says** (P_H **says** $\mathcal{C}$).

Such a derivation is possible if worldview $\omega(P_L)$ contains the appropriate beliefs.

> **CAL Requirements to Implement a Principal.** For a principal P_L to implement a principal P_H their worldviews must satisfy
>
> (i) $\mathcal{C} \in \omega(P_H) \implies$ "P_H **says** $\mathcal{C}$" $\in \omega(P_L)$
>
> (ii) "P_L **speaksfor** P_H" $\in \omega(P_H)$. $\square$

Requirement (i) implies P_L **says** (P_H **says** $\mathcal{C}$) holds whenever P_H **says** $\mathcal{C}$ does, which enables P_L to issue credentials attributing a belief $\mathcal{C}$ to P_H if P_H holds belief $\mathcal{C}$—even if P_L might not itself hold belief $\mathcal{C}$. Requirement (ii), which is equivalent to P_H **says** (P_L **speaksfor** P_H), suffices to derive P_H **says** $\mathcal{C}$ from P_L **says** (P_H **says** $\mathcal{C}$), as demonstrated by the following derivation tree.

$$\textsc{imp-e:}\frac{P_L\ \textbf{says}\ (P_H\ \textbf{says}\ \mathcal{C}),\quad \textsc{deleg-e:}\dfrac{\textsc{hand-off:}\dfrac{P_H\ \textbf{says}\ (P_L\ \textbf{speaksfor}\ P_H)}{P_L\ \textbf{speaksfor}\ P_H}}{P_L\ \textbf{says}\ (P_H\ \textbf{says}\ \mathcal{C}) \Rightarrow P_H\ \textbf{says}\ (P_H\ \textbf{says}\ \mathcal{C})}}{\textsc{says-e:}\dfrac{P_H\ \textbf{says}\ (P_H\ \textbf{says}\ \mathcal{C})}{P_H\ \textbf{says}\ \mathcal{C}}}$$

For example, if we use *HW.BOOTMGR* to name the principal corresponding to a boot loader *BOOTMGR* being executed on a processor *HW*, then SUBPRIN allows *HW.BOOTMGR* **says** C to be derived from

$$HW \textbf{ says } (HW.BOOTMGR \textbf{ says } C)$$

using the derivation tree given above. If execution of *BOOTMGR* loads and transfers control to an operating system *OS*, then any subsequent execution could be identified either with *HW.BOOTMGR.OS* or with *HW.OS*. The difference is that worldview $\omega(HW.BOOTMGR.OS)$ incorporates $Init_{HW.BOOTMGR}$, which comprises beliefs about the boot loader, whereas $\omega(HW.OS)$ does not. So worldview $\omega(HW.BOOTMGR.OS)$ is a more accurate abstraction for what is executing, supports more credentials, and enables more actions than $\omega(HW.OS)$ does.[7]

In order for principal P to implement a subprincipal $P.\eta$, the hard part is satisfying requirement (i) of CAL Requirements to Implement a Principal. It entails ensuring that P was endowed with all of the beliefs necessary to simulate every subprincipal $P.\eta$ being implemented by P. But no effort is required to satisfy requirement (ii), because (due to SUBPRIN) P_L **speaksfor** P_H directly follows from using P as P_L and using subprincipals $P.\eta$ (with different values of η) as the instances of P_H.

Group Principals. A *group principal* is defined by enumerating the finite set of principals that are its *constituents*. Different types of group principals combine the worldviews of their constituents in different ways before computing the deductive closure required by Conservative Approximation for Worldviews (page 216).

Conjunctive Group Principals. Worldview $\omega(P_G^\wedge)$ for a *conjunctive* group principal $P_G^\wedge$ constructed from constituents $G = \{P_1, ..., P_m\}$ is the deductive closure obtained from the intersection of each constituent's worldview:

$$\omega(P_G^\wedge) \quad = \quad cl_{\mathrm{CAL}}(P_G^\wedge\ , \bigcap_{P \in G} \omega(P)).$$

It is tedious, but not difficult, to prove

$$cl_{\mathrm{CAL}}(P_G^\wedge\ , \bigcap_{P \in G} \omega(P)) \quad = \quad \bigcap_{P \in G} \omega(P), \tag{9.21}$$

which implies that "conjunctive group" is the right name—$P_G^\wedge$ holds a belief if every constituent does:

$$\wedge\text{-GROUP-SAYS-I:}\ \frac{P_i \textbf{ says } \mathcal{C},\ \text{for every } P_i \in G}{P_G^\wedge \textbf{ says } \mathcal{C}}$$

[7]It is not uncommon to abbreviate the name of a subprincipal by omitting a prefix that can be inferred by readers. So we might simply write "*OS*" when readers can infer that *HW.BOOTMGR.OS* is meant or when it doesn't matter whether *HW.BOOTMGR.OS* or *HW.OS* is meant.

In addition, from (9.21) and a bit of set theory we can deduce

$$\omega(P_G^{\wedge}) \quad \subseteq \quad \omega(P) \quad \text{for } P \in G \tag{9.22}$$

which suggests the following CAL inference rule:

$$\wedge\text{-GROUP-DELEG:}\frac{}{P_G^{\wedge}\ \mathbf{speaksfor}\ P} \quad \text{for } P \in G.$$

By combining $\wedge$-GROUP-DELEG with derivation tree (9.16), we obtain derived inference rule

$$\wedge\text{-GROUP-SAYS-E:}\frac{P_G^{\wedge}\ \mathbf{says}\ \mathcal{C}}{P\ \mathbf{says}\ \mathcal{C}} \quad \text{for } P \in G$$

asserting each constituent holds any belief that the conjunctive group principal holds.

Disjunctive Group Principals. Worldview $\omega(P_G^{\vee})$ for a *disjunctive* group principal $P_G^{\vee}$ constructed from constituents $G = \{P_1, ..., P_m\}$ is the deductive closure obtained from the union of each constituent's worldview.

$$\omega(P_G^{\vee}) \quad = \quad cl_{\text{CAL}}(P_G^{\vee}, \bigcup_{P \in G} \omega(P)) \tag{9.23}$$

The definition of deductive closure $cl_{\text{CAL}}(P, B)$ given in Conservative Approximation for Worldviews implies that $B \subseteq cl_{\text{CAL}}(P, B)$, so we conclude

$$\omega(P) \quad \subseteq \quad \bigcup_{P' \in G} \omega(P') \quad \subseteq \quad \omega(P_G^{\vee}) \quad \text{for } \ P \in G, \tag{9.24}$$

which suggests the following CAL inference rule.

$$\vee\text{-GROUP-DELEG :}\frac{}{P\ \mathbf{speaksfor}\ P_G^{\vee}} \quad \text{for } P \in G$$

By combining $\vee$-GROUP-DELEG with derivation tree (9.16), we obtain a derived inference rule:

$$\vee\text{-GROUP-SAYS-I:}\frac{P\ \mathbf{says}\ \mathcal{C}}{P_G^{\vee}\ \mathbf{says}\ \mathcal{C}} \quad \text{for } P \in G$$

Thus, if a constituent holds some belief then the disjunctive group principal $P_G^{\vee}$ will hold that belief.

Definition (9.23) for $\omega(P_G^{\vee})$, however, implies that a disjunctive group principal can hold beliefs that no constituent holds, because logical consequences from the combined beliefs of different constituents are included in the deductive closure. As an example, suppose $G = \{P_1, P_2\}$ and $\mathcal{C}' \notin \omega(P_1) \cup \omega(P_2)$. Further, suppose

$$\mathcal{C} \in \omega(P_1) \qquad\qquad \text{“}\mathcal{C} \Rightarrow \mathcal{C}'\text{”} \in \omega(P_2)$$

hold, so that from (9.24) we conclude

$$\mathcal{C} \in \omega(P_G^{\vee}) \qquad\qquad \text{“}\mathcal{C} \Rightarrow \mathcal{C}'\text{”} \in \omega(P_G^{\vee})$$

and therefore

$$P_G^{\vee}\ \textbf{says}\ \mathcal{C} \qquad\qquad P_G^{\vee}\ \textbf{says}\ \mathcal{C} \Rightarrow \mathcal{C}'$$

hold. An instance of derivation tree (9.15) now serves as support for sequent

$$P_G^{\vee}\ \textbf{says}\ \mathcal{C},\ P_G^{\vee}\ \textbf{says}\ \mathcal{C} \Rightarrow \mathcal{C}' \ \ \vdash_{\text{CAL}} \ \ P_G^{\vee}\ \textbf{says}\ \mathcal{C}'.$$

Clause (ii) of Conservative Approximation for Worldviews thus implies that $\mathcal{C}' \in \omega(P_G^{\vee})$ holds. Yet $\mathcal{C}'$ does not appear in either $\omega(P_1)$ or $\omega(P_2)$. Disjunctive group principal $P_G^{\vee}$ holds a belief ($\mathcal{C}'$) that none of its constituents do—with a disjunctive group principal, the whole is greater than the sum (union) of its parts.

9.5 Accountability with Constructive Logics

A proof that some request satisfies a guard's goal formula ought to identify which principal to hold accountable for each belief involved in the authorization decision. Not all logics support such transparency of accountability. CAL does, and it is worth understanding how.

Derivations in constructive logics necessarily identify all of the evidence needed to reach a conclusion, in contrast to derivations in classical logics which may not. As an illustration, classical logics often have an inference rule that, from no premises, concludes a formula $\mathcal{F}$ is either *true* or *false*:

$$\textsc{excl-mid}^{*}\!: \frac{}{\mathcal{F} \vee \neg\mathcal{F}}$$

Conclusion $\mathcal{F} \vee \neg\mathcal{F}$ might hold because $\mathcal{F}$ holds or because $\neg\mathcal{F}$ holds—EXCL-MID* does not require a premise to distinguish which, so accountability for conclusion $\mathcal{F} \vee \neg\mathcal{F}$ is lost in logics that contain inference rule EXCL-MID*.

Now consider a derivation tree that uses EXCL-MID* to derive $\mathcal{G}$ by a form of case analysis.

$$\textsc{or-e}\!: \frac{\textsc{imp-i}(\lambda)\!: \dfrac{\dfrac{\dfrac{\boxed{\lambda\!: \mathcal{F}}}{\vdots}}{\mathcal{G}}}{\mathcal{F} \Rightarrow \mathcal{G}}, \quad \textsc{imp-i}(\lambda')\!: \dfrac{\dfrac{\dfrac{\boxed{\lambda'\!: \neg\mathcal{F}}}{\vdots}}{\mathcal{G}}}{\neg\mathcal{F} \Rightarrow \mathcal{G}}, \quad \textsc{excl-mid}^{*}\!: \dfrac{}{\mathcal{F} \vee \neg\mathcal{F}}}{\mathcal{G}} \qquad (9.25)$$

This derivation tree does not depend on whether it is $\mathcal{F}$ or $\neg\mathcal{F}$ that holds. Therefore, the derivation tree does not indicate whether $\mathcal{F}$ or $\neg\mathcal{F}$ is accountable for $\mathcal{G}$. That lack of accountability is unacceptable as a basis for an authorization decision that might later be audited. So logics that incorporate inference rule EXCL-MID* do not exhibit transparency of accountability we seek for logics intended to support credentials-based authorization.

When using CAL for a guard involving a goal formula $\mathcal{G}$ that is derived differently for the case $\mathcal{F}$ holds than for the case where $\neg\mathcal{F}$ holds, an access request would have to be accompanied by one of the two possible derivation trees

$$\begin{array}{c} \mathcal{F} \\ \hline \vdots \\ \hline \mathcal{G} \end{array} \qquad\qquad \begin{array}{c} \neg\mathcal{F} \\ \hline \vdots \\ \hline \mathcal{G} \end{array}$$

depending on whether it is $\mathcal{F}$ or $\neg\mathcal{F}$ that holds. The uncanceled assumption in each derivation tree indicates whether $\mathcal{F}$ or $\neg\mathcal{F}$ should be held accountable for $\mathcal{G}$. So the derivation tree exhibits the transparency of accountability needed for subsequent audit of authorization decisions.

Although the inference rules for any constructive logic necessarily will exhibit transparency of accountability, the choice of inference rules for constructive logics is actually driven by a somewhat different concern—support for reasoning about interpretations that have incompletely characterized states (as opposed to reasoning from incomplete information about interpretations that are completely characterized states). CAL interpretations, for instance, are constructed using partial states. A partial state could omit information necessary for knowing whether $\mathcal{F}$ or $\neg\mathcal{F}$ holds, so $\mathcal{F} \vee \neg\mathcal{F}$ might not be satisfied in a partial state. Since there can be interpretations where $\mathcal{F} \vee \neg\mathcal{F}$ is not satisfied, that formula is not valid. EXCL-MID* would thus not be sound, so it is not an inference rule of CAL or, by analogous reasoning, other constructive logics. More generally, inference rules for reasoning about incompletely characterized states must make explicit the evidence they need for a deduction, because they work only from what has become known to a reasoning agent. Transparency of accountability follows from that.

9.6 Credential Implementations

A credential that conveys P **says** $\mathcal{C}$ is worthless unless the recipient has some basis to trust that the credential was not forged or altered and, therefore, $\mathcal{C} \in \omega(P)$ was *true* when the credential was created.[8] How we implement such *credential integrity* depends on what assumptions hold about the environment and about principals.

[8]Recall, however, that beliefs in $\omega(P)$ are not themselves required to be *true*, and a compromised process might well facilitate attacks by holding beliefs that are *false*.

9.6.1 Credential Integrity from Digital Signatures

A public key K can be considered a CAL principal if we define worldview $\omega(K)$ for it. We choose $\omega(K)$ to be the smallest set that satisfies Conservative Approximation for Worldviews with $Init_K$ being the set of beliefs $\mathcal{C}$ for which a k-signed bit string

$$\mathcal{S}_k(\text{“}K \textbf{ says } \mathcal{C}\text{”}) \tag{9.26}$$

exists, where k is the private key corresponding to public key K. Notice, because the value of K appears in (9.26), holders of this signed bit string always have access to the public key needed to check that the bit string has not been forged or altered.

We have that $\mathcal{C} \in \omega(K)$ holds for every instance of (9.26) because of the way $\omega(K)$ is defined. So (9.26) satisfies credential integrity when it is interpreted as a credential conveying

$$K \textbf{ says } \mathcal{C}. \tag{9.27}$$

We now show that an instance of (9.26) also can be interpreted as a credential that conveys

$$P_K \textbf{ says } \mathcal{C} \tag{9.28}$$

provided P_K is the only principal having knowledge of private key k corresponding to public key K.

If P_K creates credential (9.26) to convey P_K **says** $\mathcal{C}$ then Credentials Foundation requires that $\mathcal{C} \in \omega(P_K)$ hold. We already established that an instance of bit string (9.26) implies that $\mathcal{C} \in \omega(K)$ holds, so we conclude $\omega(K) \subseteq \omega(P_K)$ holds. Thus, we have that K **speaksfor** P_K is sound, which enables (9.28) to be derived from (9.27) by using CAL inference rule DELEG-E. So (9.26) serves as a credential for conveying (9.28) if P_K is the only principal with knowledge of the private key k that corresponds to a public key K.

Use of a digital signature scheme to implement credentials does have some limitations. Public-private key pairs are time-consuming to generate. Also, digital signatures are expensive to create and to validate, and they are not short.[9] The most significant limitation, however, is that only certain types of principals are capable of generating k-signed bit strings and of keeping a private key secret. Thus, only certain types of principals satisfy the assumptions we require for P_K. Special-purpose cryptographic co-processors satisfy these requirements, as can privileged system software running on ordinary processors if access to memory is properly controlled. The memory of an ordinary process, however, cannot be kept secret from the privileged system software that implements processes and manages their memory. So an ordinary process cannot store a private key and issue credentials without also trusting that the underlying system software will not issue credentials that cause Credentials Foundation to be violated.

[9]If RSA is used to generate digital signatures then 2048-bit or longer private keys are recommended. The digital signature generated using a 2048-bit private key is approximately 2048 bits long.

9.6.2 Credential Integrity from Hashes

Cryptographic hash $\mathcal{H}(b)$ of a bit string b can be interpreted as a name that embodies the entire contents of b: A change to even one bit in b yields (with very high probability) an unpredictably different value for $\mathcal{H}(b)$, hence an unpredictably different name. Names inextricably linked to what they denote can serve as names for principals that are inextricably linked to sets of beliefs. Below, we use this observation and show how cryptographic hash functions can implement credential integrity for certain applications.

Assume some well known invertible bit string representation $rep(\mathcal{C})$ for any CAL formula $\mathcal{C}$, so that $rep^{-1}(rep(\mathcal{C})) = \mathcal{C}$. We also postulate an encoding $b \triangleright rep(\mathcal{C})$ for unambiguously augmenting an arbitrary bit string b with these representations, and (since the encoding is unambiguous) extend $rep^{-1}(\cdot)$ to recover $\mathcal{C}$ from $b \triangleright rep(\mathcal{C})$:

$$rep^{-1}(b \triangleright rep(\mathcal{C})) \;\; = \;\; \mathcal{C}$$

We interpret a bit string $b \triangleright rep(\mathcal{C})$ as a credential that conveys CAL formula

$$\mathcal{H}(b \triangleright rep(\mathcal{C})) \textbf{ says } \mathcal{C} \tag{9.29}$$

thereby attributing belief $\mathcal{C}$ to a principal named by a cryptographic hash. And we define worldview $\omega(\mathcal{H}(b \triangleright rep(\mathcal{C})))$ to be the smallest set of beliefs that satisfies Conservative Approximation for Worldviews, where $Init_{\mathcal{H}(b \triangleright rep(\mathcal{C}))}$ is the set of all beliefs specified by CAL formulas $\mathcal{C}'$ satisfying

$$\mathcal{H}(b \triangleright rep(\mathcal{C})) \;\; = \;\; \mathcal{H}(b \triangleright rep(\mathcal{C}')). \tag{9.30}$$

We might have hesitations about adopting a definition for the worldview of $b \triangleright rep(\mathcal{C})$ that, besides containing belief $\mathcal{C}$, includes a seemingly random set of other beliefs—namely, those beliefs $\mathcal{C}'$ satisfying (9.30). However, $\mathcal{C}'$ cannot be attributed to principal $\mathcal{H}(b \triangleright rep(\mathcal{C}))$ unless there were an efficient way, given $b \triangleright rep(\mathcal{C})$, to obtain CAL formula $\mathcal{C}'$ satisfying (9.30). Since $\mathcal{H}(\cdot)$ is a cryptographic hash function, brute-force enumeration to find such a $\mathcal{C}'$ is infeasible, and the Weak Collision Resistance property[10] for cryptographic hash functions rules out the existence of any faster means for obtaining $\mathcal{C}'$ from $b \triangleright rep(\mathcal{C})$

The existence of fast algorithms to compute cryptographic hashes means that the scheme just outlined is an attractive way to achieve credential integrity. The benefits, though, are offset by inflexibility regarding what beliefs can be attributed to the principal having any given name—belief $\mathcal{C}$ can be attributed only to the principal having name $\mathcal{H}(b \triangleright rep(\mathcal{C}))$. Moreover, change the set[11] of beliefs, and the name of the principal to which they are being attributed is likely to change. So hash-based credentials are well suited only for applications

[10]Weak Collision Resistance asserts that, given a bit string b, it is infeasible to compute another bit string b' where $\mathcal{H}(b) = \mathcal{H}(b')$ holds.

[11]A set comprising beliefs $\mathcal{C}_1$, $\mathcal{C}_2$, ..., $\mathcal{C}_n$ is equivalent to a single belief $\mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \cdots \wedge \mathcal{C}_n$.

where (i) arbitrary principal names can be accommodated and (ii) the set of beliefs being associated with that principal never changes.

One such application arises in connection programs. A program is often viewed as a principal holding a fixed sets of beliefs for enabling access to some data that program manages. Suppose bit string pgm is the binary representation for such a program. We thus can use $\texttt{pgm} \rhd \mathit{rep}(\mathcal{C})$ for attributing belief $\mathcal{C}$ to pgm. To execute pgm, a user U would first invoke a system-provided action that declares trust in the principal $\mathcal{H}(\texttt{pgm} \rhd \mathit{rep}(\mathcal{C}))$ associated with whatever *executable* is represented by bit string $\texttt{pgm} \rhd \mathit{rep}(\mathcal{C})$. Execution by U of this system-provided action is, by convention, interpreted to convey

$$U \ \textbf{says} \ (\mathcal{H}(\texttt{pgm} \rhd \mathit{rep}(\mathcal{C})) \ \textbf{speaksfor} \ U). \tag{9.31}$$

After U declares its trust in principal $\mathcal{H}(\texttt{pgm} \rhd \mathit{rep}(\mathcal{C}))$, program pgm then can be invoked by U. Given (9.31), the CAL HAND-OFF inference rule will attribute to U actions that pgm performs as well as attributing belief $\mathcal{C}$ to U. So beliefs held by U plus belief $\mathcal{C}$ can be required in goal formulas used to authorize requests made by U's execution of pgm.

The benefit of the approach becomes clear when we consider what happens if an attacker substitutes a compromised version pgm′ for pgm and somehow fools user U into invoking executable $\texttt{pgm}' \rhd \mathit{rep}(\mathcal{C})$ after U has declared trust in $\texttt{pgm} \rhd \mathit{rep}(\mathcal{C})$ through (9.31). Because

$$\mathcal{H}(\texttt{pgm} \rhd \mathit{rep}(\mathcal{C})) \neq \mathcal{H}(\texttt{pgm}' \rhd \mathit{rep}(\mathcal{C}))$$

likely holds, $\texttt{pgm}' \rhd \mathit{rep}(\mathcal{C})$ attributes $\mathcal{C}$ to a different principal than specified by U in delegation (9.31). So if U is fooled into invoking pgm′ then the actions pgm′ performs and belief $\mathcal{C}$ will not be attributed to U. A goal formula that requires $\mathcal{C}$ to be attributed to U would cause the guard to block requests that the compromised program issues.

9.6.3 Kernel Support for Credential Integrity

The final approach we explore for credential integrity employs an operating system kernel. The kernel would thus be part of the trusted computing base, though it is likely to be already present for other reasons.[12] Our implementation depends on the following guarantees about processes that the kernel implements.

- Each process P can read or write memory it owns but is denied access to all other memory.
- The kernel can read or write memory it owns.[13]

[12]In a networked system, however, the scheme we outline would require the kernel on every host be trusted by all of the other hosts—an assumption that might be plausible for an intranet operated by a single enterprise but hard to defend for an open internet.

[13]Often, the kernel is authorized to read and write memory that any process owns, too. The approach we describe to credential integrity is unaffected by allowing the additional access.

```
AddBelief: operation( blf : formula )
        $WV[\,\mathbf{caller}()\,] := WV[\,\mathbf{caller}()\,] \cup \{blf\}$
        end AddBelief

DelBelief: operation( blf : formula )
        $WV[\,\mathbf{caller}()] := WV[\,\mathbf{caller}()\,] - \{blf\}$
        end DelBelief

CheckCred: function( cred : credential ) returns (boolean)
        $CheckCred := cred \in (\bigcup_p \{p \textbf{ says } blf \mid blf \in WV[p]\})$
        end CheckCred

QueryCred: function( q : query ) returns (set of credential)
        $QueryCred := q(\bigcup_p \{p \textbf{ says } blf \mid blf \in WV[p]\})$
        end QueryCred
```

Figure 9.6: In-Kernel Credentials Database Implementation

- The identity of the process invoking a system call is available to the kernel code servicing that invocation.

If every principal is implemented by a separate process then the first guarantee implies that credentials stored in memory the kernel owns cannot be forged or corrupted by any principal, whereas the second and third guarantees facilitate having system calls be the sole means by which processes can alter the sets of credentials being stored by the kernel. We now turn to the details.

For each principal P, a table entry $WV[P]$ is stored in kernel-owned memory. $WV[P]$ contains CAL formulas for some subset of the beliefs that P holds:

$$WV[P] \subseteq \omega(P) \tag{9.32}$$

Initializing $WV[P]$ to $\emptyset$ makes (9.32) hold.

Deletion of an element from $WV[P]$ cannot invalidate (9.32), although we might want to restrict this operation to the authority on $\omega(P)$—likely, principal P—or to operating system code that manages storage consumed by $WV[P]$. By deleting elements from $WV[P]$, a principal P can accommodate changes to its worldview $\omega(P)$, something that is explored at length in §9.8.

Adding a CAL formula to $WV[P]$ cannot invalidate (9.32) if P is the only principal permitted to add credentials to $WV[P]$, because Credentials Foundation implies that $\mathcal{C} \in \omega(P)$ will hold when P attempts to add a credential that conveys a belief $\mathcal{C}$.

Figure 9.6 sketches implementations for system calls *AddBelief* to add a credential, *DelBelief* to delete a credential, *CheckCred* to determine whether a specific credential is being stored, and *QueryCred* to retrieve the subset of all credentials satisfying some given query q. The code for *AddBelief* and *DelBelief* is written using a **caller**() primitive, which returns the name of the invoking process (hence, the name of an associated principal). For simplicity, we assume that queries q passed to *QueryCred* are functions that return some subset of their input (i.e, those credentials satisfying the criteria specified in the query). Notice, *AddBelief* and *QueryCred* can together be used to pass a credential from one process to another.[14]

An in-kernel table like WV also can provide a clean interface for processes to learn about states of operating system abstractions—for example, the amount of free space available in the file system or the processor load. System state is portrayed as beliefs attributed to a special principal, OS; *CheckCred* and *QueryCred* provide processes with access. Beliefs attributed to OS would probably not actually be stored in WV, though, to avoid the considerable overhead needed to keep such constantly-changing information current. Rather, beliefs attributed to OS would be generated, as needed, whenever $WV[OS]$ is accessed.

Kernel Cache for Derivation Trees. A single derivation tree will often serve as support for multiple requests. For example, when enforcing discretionary access control for a file system, all requests by a given process to read a specific file require support for the same guard sequent (which establishes that the requester is among the principals authorized to read the file). A single derivation tree thus can be reused. Opportunities for reuse of derivation trees are present with other authorization policies, too.

System performance suffers unnecessarily when reuse of derivation trees is not supported. Resending a derivation tree consumes bandwidth, storing copies consumes memory, and rechecking a tree consumes processor cycles. Therefore, a single shared cache for storing checked derivation trees is attractive. Moreover, such a cache simplifies guard programming if the cache supports a search operation that determines whether a derivation tree being stored (i) has a conclusion that matches some specified goal formula and (ii) has assumptions that all are present in some trusted credentials database. Derivation trees would now no longer need to accompany each access request. A guard simply queries the cache when checking the guard sequent for a given request being authorized; and each process, before making an access request, ensures that a suitable derivation tree appears in the cache.

Figure 9.7 sketches an implementation of such a cache; it uses the credentials database implementation of Figure 9.6.

AddDervTree(dt) checks whether dt is a derivation tree and, if so, stores it in *DervTrees*. Boolean function *isDerivTree*(dt) checks that dt satisfies Derivation Tree Formal Definition (page 208) by verifying that each node of derivation tree dt is an instance of the indicated CAL inference rule.

[14]The implementation of Figure 9.6 does not restrict which processes are authorized to retrieve a given credential by invoking *QueryCred*, but code to support such functionality is easily added to better approximate the semantics of IPC primitives.

```
var $DervTrees$: set of derivation tree initial($\emptyset$)

$AddDrvTree$: operation( $dt$ : derivation tree )
                  if $isDerivTree(dt)$ then $DervTrees$:= $DervTrees \cup \{dt\}$
                  end $AddDerivTree$

$CheckConcl$: operation( $f$ : formula ) returns (boolean)
                  var $dt$ : derivation tree
                  if $dt \in DervTrees$ such that
                        $Conc(dt) = f$
                        $\wedge$ for all $a \in Asmpts(dt)$: $CheckCred(a)$
                    then $CheckConcl := true$
                    else $CheckConcl := false$
                  end $CheckConcl$
```

Figure 9.7: In-Kernel Derivation-Tree Database Implementation

CheckConcl(f) returns *true* whenever there is some derivation tree dt stored in *DervTrees* that satisfies

- conclusion $Conc(dt)$ is CAL formula f, and
- each CAL formula C_i in set $Asmpts(dt)$ of uncanceled assumptions is in the credentials database.

Figure 9.7 offers no operation to delete a derivation tree, because system execution never falsifies $isDerivTree(dt)$ for any derivation tree dt stored in *DervTrees*. Deletion of a credential from the credentials database, however, can render a derivation tree dt irrelevant if that credential is an uncanceled assumption of dt. No harm comes from storing derivation trees that become irrelevant, but also no harm comes from deleting from *DervTrees* relevant or irrelevant derivation trees—say, to control storage costs—because a derivation tree can be reloaded if ever it is needed but no longer present.

9.7 Guard and Credential Pragmatics

Naming. The designer of a guard must decide what sources to trust for information about current and past states. Presumably, a guard would trust predicate evaluations that it performs itself or that an operating system kernel (which the guard must trust anyway) performs on its behalf. Other components might have to be trusted, too, because it is unlikely that every principal would be able to evaluate every predicate, due to constraints imposed by locality and/or confidentiality. Arguably, a large part of designing a secure system is concerned with aligning what must be trusted with what can be trusted. Credentials-based authorization helps focus on these design choices by having each credential explicitly bind the name of principal to the belief that credential conveys, thereby surfacing what is being trusted.

CAL is agnostic about predicate and function names, assuming only that all principals assign the same meaning to each name. One approach is to standardize the name and meaning (including an evaluation scheme) of all predicates and functions that guards may use. Implicit in such a solution would have to be some means for finding a compliant implementation for each predicate or function. Hierarchical naming, for example, could be used to construct names that encode the identity of the principal that certifies compliant implementations.

Requested Operations as Beliefs. Goal Formula Determination (step (1) of Guard Operation, page 206) makes it redundant for a goal formula $\mathcal{G}_\Theta$ authorizing operation Θ by a principal P to include

$$P \textbf{ says } \Theta \tag{9.33}$$

as a conjunct. That redundancy, nevertheless, can be helpful. It forestalls a request from being erroneously authorized because the wrong goal formula was selected due to bugs in the code that implements Goal Formula Determination or due to an operator mistakenly installing the goal formula for a different operation.

In addition, a modest strengthening of (9.33) can allow guards to defend against replay attacks. Having (9.33) be a conjunct of $\mathcal{G}_\Theta$ binds an invocation of Θ to a belief that P holds. To eliminate the possibility of replay attacks,

- each specific invocation of Θ would be linked to a distinct belief (instead of all invocations being linked to a single belief), and
- a correspondingly stronger version of (9.33) would be included as a conjunct of goal formula $\mathcal{G}_\Theta$, thereby causing the guard to check that a distinct belief is being used each time Θ is authorized.

In the obvious implementation, each principal P that invokes operation Θ replaces its belief Θ by beliefs Θ_1, Θ_2, ..., where P **says** Θ_i authorizes the i^{th} invocation of operation Θ by P. The guard then maintains an integer array $\mathit{last}[P]$ that records the index labeling the last Θ-invocation by P the guard authorized, and goal formula $\mathcal{G}_\Theta$ includes the following strengthening of (9.33):

$$P \textbf{ says } \Theta_i \quad \wedge \quad i > \mathit{last}[P]$$

An alternative defense against replay attacks is to include the simpler (9.33) as a conjunct of guard formula $\mathcal{G}_\Theta$ but ensure that P and the guard are the only components that ever have access to a credential that conveys (9.33). The communications channel between P and the guard would thus need to be confidentiality-protected. Attackers now lack a credential for satisfying (9.33), so the credentials accessible to an attacker attempting a replay attack would be insufficient to satisfy guard formula $\mathcal{G}_\Theta$.

Goal Formula Templates. The decision to authorize a request often will depend on the identity of the principal making the request, properties of arguments being passed to the requested operation, and/or other parameters of credentials accompanying the request. A *goal formula template* can succinctly specify such an authorization policy for a class of requests. The template is a CAL formula written in terms of one or more *template parameters*, which we typeset here as underlined, sans-serif font (i.e., $\underline{\mathsf{a}}$, $\underline{\mathsf{b}}$, ...); an actual goal formula is generated by replacing the template parameters with information found on the credentials that accompany a request.

For example, goal formula template

$$\begin{aligned} & \underline{\mathsf{P}}\ \textbf{says}\ \texttt{FreeMem}(\underline{\mathsf{strt}}, \underline{\mathsf{end}}) \\ \wedge\ & \underline{\mathsf{P}}\ \textbf{speaksfor}\ \texttt{OS} \\ \wedge\ & 0 \leq \underline{\mathsf{strt}} < \underline{\mathsf{end}} \leq MAX \end{aligned} \tag{9.34}$$

introduces template parameters $\underline{\mathsf{P}}$, $\underline{\mathsf{strt}}$, and $\underline{\mathsf{end}}$. Goal formulas generated from this template enforce an authorization policy that restricts the request source (*viz.* $\underline{\mathsf{P}}$) to being a principal that speaks for principal `OS` (presumably, the operating system) and restrict the arguments (*viz.* $\underline{\mathsf{strt}}$ and $\underline{\mathsf{end}}$) to the requested `FreeMem` operation.

A set of credentials could admit a number of possible instantiations for template parameters, each generating a different goal formula. For example,

$$\begin{aligned} & \texttt{Editor}\ \textbf{says}\ \texttt{FreeMem}(1024, 2048) \\ \wedge\ & \texttt{Editor}\ \textbf{speaksfor}\ \texttt{OS} \\ \wedge\ & 0 \leq 1024 < 2048 \leq MAX \end{aligned}$$

is among the goal formulas that might be generated from goal formula template (9.34) for a request accompanied by credentials $\texttt{C}_1$ and $\texttt{C}_2$ conveying

$$\begin{aligned} \mathcal{M}(\texttt{C}_1):&\quad \texttt{Editor}\ \textbf{says}\ \texttt{FreeMem}(1024, 2048) \\ \mathcal{M}(\texttt{C}_2):&\quad \texttt{OS}\ \textbf{says}\ (\texttt{Editor}\ \textbf{speaksfor}\ \texttt{OS}). \end{aligned}$$

A finite number of goal formulas can be generated given a goal formula template and finite set of credentials that accompany some request. The guard authorizes a request if any one of the generated goal formulas yields a guard sequent for which there is CAL support. And to reduce the cost of generating all of the possible goal formulas (and constructing or checking a CAL derivation tree for each), a guard might expect requests to be accompanied with proposed instantiations for template parameters.

Getting Support for Guard Sequents. A *universal guard* would take as inputs

- any goal formula $\mathcal{G}$ and
- CAL formulas $\mathcal{C}_1$, $\mathcal{C}_2$, ..., $\mathcal{C}_n$ conveyed by a set of credentials.

It would construct and output a derivation tree, if one exists, having conclusion $\mathcal{G}$ and having all of its uncanceled assumptions appearing in $\mathcal{C}_1$, $\mathcal{C}_2$, ..., $\mathcal{C}_n$; otherwise, some distinguished value unsupported would be output.

Having a universal guard would be a boon to implementing credentials-based authorization. Unfortunately, Gödel's first incompleteness theorem (a classical result in formal logic) implies that universal guards cannot exist. Gödel proved that no axiomatization of arithmetic can admit the kind of automated deduction that universal guards would provide. And CAL or any logic for reasoning about integers or non-trivial data structures must be extending an axiomatization of arithmetic.

The infeasibility of universal guards, however, does not preclude implementations of Authorization Decision (step 4, page 206) for guards that require an access request to include

(i) credentials having some pre-specified form, and/or

(ii) a derivation tree or parts thereof having pre-specified uncanceled assumptions and conclusion.

Given (i), a guard could be programmed to generate the required derivation tree from the client-provided credentials. Clients, however, would have to provide credentials in exactly the right form or risk having their access requests be denied. With (ii), the guard would grant an access only after checking that client-supplied derivation trees satisfy Derivation Tree Formal Definition (page 208). Such checking is feasible, because derivation trees are finite and correct applications of CAL inference rules can be verified mechanically.

Note, use of (i) or (ii) implies that changing the goal formula for a deployed guard could require finding and updating all principals that might submit requests to that guard. Approach (ii) also requires disclosing the goal formula to clients, yet there could be reasons for a goal formula to be kept secret—for example, the same kind of request from different principals might need to satisfy different conditions.

Guard for a File System. To illustrate, we show how a guard for enforcing discretionary access control in a file system `FileSys` might employ (i) and (ii). Such a guard should allow an access request to proceed if that request is from the owner of a file or it is from any principal that has been delegated access from a principal that has access. With this in mind, we choose

$$\texttt{FileSys}\ \textbf{says}\ \Theta(\underline{\mathsf{f}}) \tag{9.35}$$

to be the goal formula template for performing Θ operations on a file $\underline{\mathsf{f}}$. The owner $own(F)$ of file F then would be issued restricted delegation

$$\texttt{FileSys}\ \textbf{says}\ (own(F)\ \textbf{speaks}\ \Theta(F)\ \textbf{for}\ \texttt{FileSys}) \tag{9.36}$$

by `FileSys`. And any principal P that has been delegated authorization for $\Theta(F)$ in turn delegates that authorization to another principal Q by issuing the

$$\text{IMP-E:}\ \dfrac{\underline{\mathsf{P}}\ \textbf{says}\ \Theta(\underline{\mathsf{f}}),\quad \text{REST-DELEG-E:}\ \dfrac{\mathcal{D}(\underline{\mathsf{f}},\underline{\mathsf{P}})\text{:}\ \dfrac{\mathcal{C}_1,\ \mathcal{C}_2,\ \ldots,\ \mathcal{C}_n}{\underline{\mathsf{P}}\ \textbf{speaks}\ \Theta(\underline{\mathsf{f}})\ \textbf{for}\ \mathit{FileSys}}}{(\underline{\mathsf{P}}\ \textbf{says}\ \Theta(\underline{\mathsf{f}})) \Rightarrow (\mathit{FileSys}\ \textbf{says}\ \Theta(\underline{\mathsf{f}}))}}{\mathit{FileSys}\ \textbf{says}\ \Theta(\underline{\mathsf{f}})}$$

Figure 9.8: Template for DAC Sequent Support

restricted delegation

$$P\ \textbf{says}\ (Q\ \textbf{speaks}\ \Theta(F)\ \textbf{for}\ P). \qquad (9.37)$$

For a principal $\mathit{own}(F)$ to perform $\Theta(F)$, the file system guard is sent a credential that conveys P **says** $\Theta(F)$ and a credential that conveys restricted delegation (9.36). These CAL formulas together suffice to authorize the $\Theta(F)$ request, because they allow goal formula (9.35) to be derived. For a principal Q that is not the owner of file F to perform $\Theta(F)$, it submits a credential that conveys Q **says** $\Theta(F)$, a set of credentials that convey $\mathcal{C}_1$, $\mathcal{C}_2$, ..., $\mathcal{C}_n$, as well as a derivation tree $\mathcal{D}(F, Q)$ that is support for sequent:

$$\mathcal{C}_1,\ \mathcal{C}_2,\ \ldots,\ \mathcal{C}_n\ \ \vdash_{\text{CAL}}\ \ Q\ \textbf{speaks}\ \Theta(F)\ \textbf{for}\ \texttt{FileSys}$$

In either case, the guard can instantiate the template in Figure 9.8 based on information accompanying the access request: The name of the file being read would be substituted for template parameter $\underline{\mathsf{f}}$, the name of the principal making the request would be substituted for template parameter $\underline{\mathsf{P}}$, and a derivation tree would substitute for the inference rule instantiation with label $\mathcal{D}(\underline{\mathsf{f}},\underline{\mathsf{P}})$. For owner P, derivation tree

$$\text{REST-HAND-OFF:}\ \dfrac{\texttt{FileSys}\ \textbf{says}\ (\mathit{own}(F)\ \textbf{speaks}\ \Theta(F)\ \textbf{for}\ \texttt{FileSys})}{P\ \textbf{speaks}\ \Theta(F)\ \textbf{for}\ \mathit{FileSys}}$$

would be substituted for $\mathcal{D}(\underline{\mathsf{f}},\underline{\mathsf{P}})$; for non-owners, derivation tree $\mathcal{D}(F, Q)$ provided in the request would be substituted for $\mathcal{D}(\underline{\mathsf{f}},\underline{\mathsf{P}})$.

9.8 Changes to Beliefs

We should expect that beliefs in worldview $\omega(P)$ would reflect the current state and environment of a principal P. Changes to $\omega(P)$ then bring two kinds of complications.

- Beliefs being attributed to P by another principal P' might no longer be held by P. Problems are avoided if (i) P informs P' before P changes its beliefs, or (ii) P' queries P before P' undertakes any action that presupposes P holds a given belief.

- Receipt by P' of a credential might convey beliefs no longer current, despite Credentials Foundation (page 216) and credential integrity. Such *stale* credentials can be eliminated by imposing restrictions on how credentials are disseminated and restrictions on what beliefs they convey.

Protocols for the first are straightforward exercises in concurrent programming, where actions by one process must be synchronized with state changes by another. So the focus in this section is on the second—restrictions to avoid stale credentials.

Use of Authorities. A credential cannot become stale after it has been deleted. Thus, a principal P can remove or modify a belief $\mathcal{C}$ in $\omega(P)$ provided P has deleted all credentials conveying P **says** $\mathcal{C}$. Without restrictions on credential propagation and storage, though, undertaking system-wide credential deletion is likely to be infeasible. Credentials first must be found, yet they might be stored anywhere in a system's files or memory—perhaps unrecognizable if application-specific formats are in use.

A similar effect to finding and deleting stale credentials is achieved by preventing principals from storing or forwarding credentials that might become stale. We call a source of credentials an *authority* if it is the sole provider of information about some set of beliefs. So a response from an authority $P_{\mathcal{C}}$ on some belief $\mathcal{C}$, by definition, both conveys whether $\mathcal{C} \in \omega(P_{\mathcal{C}})$ is *true* and cannot be used to convince other clients of whether $\mathcal{C} \in \omega(P_{\mathcal{C}})$ is *true*.

The response from some authority $P_{\mathcal{C}}$ typically will indicate only whether $\mathcal{C} \in \omega(P_{\mathcal{C}})$ did hold when the response was generated. To infer that $\mathcal{C} \in \omega(P_{\mathcal{C}})$ still holds after the response was generated requires reasoning and/or restrictions. An authority $P_{\mathcal{C}}$ might be designed to delay changing a belief $\mathcal{C}$ for at least T seconds after responding to any client request about $\mathcal{C}$, where T is some publicly known value. In effect, credentials issued by $P_{\mathcal{C}}$ expire T seconds after the response they convey was generated. A client that cannot determine exactly when some given response was generated nevertheless can conservatively budget for the credential to expire T seconds after that client's request was submitted.[15]

The protocol executed by a client P seeking to determine whether authority $P_{\mathcal{C}}$ holds a belief $\mathcal{C}$ is a straightforward query-response over some integrity-protected and authenticated communications channel $Ch^{P}_{P_{\mathcal{C}}}$ (say). The authentication and integrity protection assumptions for channel $Ch^{P}_{P_{\mathcal{C}}}$ imply

$$Ch^{P}_{P_{\mathcal{C}}} \textbf{ speaksfor } P_{\mathcal{C}}. \tag{9.38}$$

and, therefore, P can derive

$$P_{\mathcal{C}} \textbf{ says } \mathcal{C}$$

from

$$Ch^{P}_{P_{\mathcal{C}}} \textbf{ says } (P_{\mathcal{C}} \textbf{ says } \mathcal{C})$$

which is what the response message received on channel $Ch^{P}_{P_{\mathcal{C}}}$ would be conveying to P.

[15]This assumes that the clocks at the client and the authority advance at similar rates.

Mechanism	Assumptions
system calls	(i) authority must be part of the operating system kernel and (ii) operating system must be trusted
system-provided IPC channels	(i) authority must be a process on the same machine as the client and (ii) operating system must be trusted.
digitally-signed messages	(i) private key is known only to authority, (ii) public key us known to all clients, and (iii) a client's query includes a globally-unique nonce, which the authority incorporates into its digitally-signed response.
MAC-protected messages	authority and each client share a (symmetric) key.

Figure 9.9: Implementation Options for Authorities

Figure 9.9 lists various mechanisms for implementing the integrity-protected and authenticated channel from an authority to a client. Note, the first mechanism listed (system calls) employs the kernel support for credential integrity discussed in §9.6.3. Nonces in the digitally-signed messages are needed to prevent principals from storing and forwarding old responses from authorities.

Weakening and Split Credentials. CAL formula $\mathcal{C} \vee \mathcal{C}'$ is *weaker* than $\mathcal{C}$—it rules out fewer interpretations and, thus, it rules out fewer worldviews. So changes to a principal's beliefs that invalidate $\mathcal{C}$ might not invalidate $\mathcal{C} \vee \mathcal{C}'$. That makes weaker formulas good candidates to convey in credentials.

A particularly useful construction is to employ one disjunct as a signal for whether another disjunct holds, creating what we call a *split credential*.

Split Credential. Receipt of a credential `C` that conveys

$$P \textbf{ says } (\neg\mathcal{B} \vee \mathcal{C}), \tag{9.39}$$

along with some basis to conclude P **says** $\mathcal{B}$, suffices for deriving P **says** $\mathcal{C}$ in CAL. Moreover, split credential `C` cannot become stale provided P holds belief $\neg\mathcal{B}$ whenever P does not hold belief $\mathcal{C}$. □

In sum, a principal P that issues a split credential conveying (9.39) is (i) free to invalidate $\mathcal{C}$ provided P also holds belief $\neg\mathcal{B}$, and (ii) free to invalidate $\neg\mathcal{B}$ provided P also holds belief $\mathcal{C}$.

The Split Credential construction replaces a credential to convey P **says** $\mathcal{C}$, which can become stale, with a weaker credential. Doing this involves introducing a new credential to convey P **says** $\mathcal{B}$. Although that new credential might itself become stale, we have complete freedom in the choice of $\mathcal{B}$; in contrast, $\mathcal{C}$ presumably would be constrained by the authorization policy that is being implemented. By choosing for $\mathcal{B}$ a belief that is available solely from some authority $P_{\mathcal{B}}$, no credentials would be created or stored for conveying P **says** $\mathcal{B}$, so none becomes stale. Authority $P_{\mathcal{B}}$ and client P might be one and the same principal. Or $P_{\mathcal{B}}$ might be a different principal that is trusted by P. Examples of the latter include a time service or the guard whose goal formula requires P **says** $\mathcal{C}$.

A common class of Split Credential constructions instantiate $\mathcal{B}$ by a belief that never changes value from *false* to *true*. For (9.39) not to be invalidated, P is obligated to hold belief $\mathcal{C}$ for some initial period that terminates after $\mathcal{B}$ first becomes *false*. Think in terms of belief $\mathcal{C}$ in (9.39) as expiring or being revoked at the instant $\mathcal{B}$ transitions from *true* to *false*.

Such a $\mathcal{B}$ is easy to construct if some variable is available that stores non-decreasing values, such as sequence numbers or time. Let *nonDecr* be that variable. And suppose changes to *nonDecr* are controlled by an authority P_A that is trusted by P, as signified through delegation:

$$P \textbf{ says } P_A \textbf{ speaks } v\colon \mathit{nonDecr} < v \textbf{ for } P$$

If, for instance, we choose $\mathit{nonDecr} < 10$ for $\mathcal{B}$ then P is obligated to hold belief $\mathcal{C}$ only until *nonDecr* is incremented so that $\mathit{nonDecr} \geq 10$ is satisfied. Thus, receipt of a split credential conveying

$$P \textbf{ says } (\mathit{nonDecr} \geq 10 \ \vee \ \mathcal{C})$$

along with information from authority P_A that implies

$$P_A \textbf{ says } \mathit{nonDecr} < 10$$

allows P **says** $\mathcal{C}$ to be derived in CAL by P.

9.9 Multi-level Security Revisited

Most descriptions of multi-level security (including §8.1 and §8.1.2) ignore the infrastructure for assigning labels to files and for assigning clearances to users. Yet that infrastructure plays a central role in determining whether an access request will be authorized. By reformulating multi-level security in terms of credentials-based authorization, we can account for the label-assignment infrastructure. The reformulation also illustrates how credentials-based authorization exposes what principals must be trusted.

Label-assignment likely would be performed by more than one *classification authority*, each with jurisdiction to assign some labels but not others.

- Specialized and sometimes secret knowledge about subject matter can be required to assign the appropriate label $\mathcal{L}(F)$ to a file F, so postulating that a single entity labels all files is not sensible.

- The expertise required for assessing trustworthiness of a human user U in order to assign a clearance $\mathcal{L}(U)$ is different than what is needed for labeling files, so people and files likely would be assigned labels by different classification authorities.

Moreover, for all the usual reasons favoring delegation of authority, we would want to have classification authorities be distinct from guards and distinct from principals that create files.

Our formulation of multi-level security in terms of credentials-based authorization presumes that each file F has owner $own(F)$ and a label $\mathcal{L}(F)$. We also postulate that any program Pgm executing for a user U is given a label $\mathcal{L}(Pgm)$ by the operating system OS, where $\mathcal{L}(Pgm) = \mathcal{L}(U)$ holds. These labels are conveyed to guards through credentials; guards mediate read and write requests accordingly.

Guards use the goal formula template

$$\begin{array}{rl} & Pgm\ \mathbf{says}\ read(\underline{\mathsf{f}}) \\ \wedge & own(\underline{\mathsf{f}})\ \mathbf{says}\ \mathcal{L}(\underline{\mathsf{f}}) = l_{\underline{\mathsf{f}}} \\ \wedge & OS\ \mathbf{says}\ \mathcal{L}(Pgm) = l_{Pgm} \\ \wedge & l_{\underline{\mathsf{f}}} \preceq l_{Pgm} \end{array} \tag{9.40}$$

to authorize a request from a program Pgm to read a file that instantiates template parameter $\underline{\mathsf{f}}$. Thus, the read request is allowed to proceed only if requester Pgm has a clearance that dominates the label on the file to be read.[16]

Beliefs about labels $\mathcal{L}(F)$ and $\mathcal{L}(Pgm)$ are in the purview of classification authorities. Those *roots of trust* are selected by whomever is responsible for establishing assurance in the labels being used. A classification authority A serves as a root of trust for labels used by a principal P if P holds a credential that conveys a restricted delegation for some set Obj_A of objects that P trusts A to label:

$$P\ \mathbf{says}\ (A\ \mathbf{speaks}\ \ \langle l, o\rangle{:}(o \in Obj_A \Rightarrow \mathcal{L}(o) = l)\ \ \mathbf{for}\ \ P) \tag{9.41}$$

This restricted delegation enables beliefs to be attributed to P if they have source A and have form "$o \in Obj_A \Rightarrow \mathcal{L}(o) = l$". So (9.41) is asserting that, according to principal A, if $o \in Obj_A$ holds then $\mathcal{L}(o)$ has some given label. For example,

$$own(F)\ \mathbf{says}\ (F \in Obj_A \Rightarrow \mathcal{L}(F) = l_F) \tag{9.42}$$

can be derived from a credential that conveys

$$A\ \mathbf{says}\ (F \in Obj_A \Rightarrow \mathcal{L}(F) = l_F)$$

from an instance (9.41) that substitutes $own(F)$ for P.[17]

[16]The goal formula for *write* is analogous, except the first conjunct would be Pgm **says** $write(\underline{\mathsf{f}})$ and the final conjunct ("no read-up") is switched to $l_{Pgm} \preceq l_{\underline{\mathsf{f}}}$ ("no write-down").

[17]The derivation tree is built using REST-HAND-OFF, REST-DELEG-E, and IMP-E.

We formalize in CAL a belief for asserting that F is among set Obj_A of objects that A is trusted by $own(F)$ to label as:

$$own(F) \textbf{ says } F \in Obj_A \tag{9.43}$$

SAYS-IMP-MP with (9.42) and (9.43) then allows us to conclude

$$own(F) \textbf{ says } \mathcal{L}(F) = l_F.$$

as required in goal formula (9.40).

A similar derivation with some (perhaps different) classification authority A' would be employed to derive a label assigned by OS to Pgm. Specifically, OS would make a restricted delegation to A':

$$OS \textbf{ says } (A' \textbf{ speaks } \langle l, p\rangle : (p \in Progs_{A'} \Rightarrow \mathcal{L}(p) = l) \textbf{ for } OS) \tag{9.44}$$

This then enables

$$OS \textbf{ says } (Pgm \in Progs_{A'} \Rightarrow \mathcal{L}(Pgm) = l_{Pgm}) \tag{9.45}$$

to be derived from a credential (presumably from A') that conveys

$$A' \textbf{ says } (Pgm \in Progs_{A'} \Rightarrow \mathcal{L}(Pgm) = l_{Pgm})$$

Coupled with a belief that OS holds

$$OS \textbf{ says } Pgm \in Progs_{A'}$$

about its root of trust for user labels, (9.45) derives

$$OS \textbf{ says } \mathcal{L}(Pgm) = l_{Pgm}$$

Putting all this together, we obtain the following guard sequent for goal formula template (9.40). It incorporates classification authorities as the roots of trust for label assignments.

$$\begin{array}{l}
Pgm \textbf{ says } read(F), \\
own(F) \textbf{ says } (A \textbf{ speaks } \langle l, o\rangle : (o \in Obj_A \Rightarrow \mathcal{L}(o) = l) \textbf{ for } own(F)), \\
own(F) \textbf{ says } F \in Obj_A, \\
A \textbf{ says } (F \in Obj_A \Rightarrow \mathcal{L}(F) = l_F), \\
OS \textbf{ says } (A' \textbf{ speaks } \langle l, p\rangle : (p \in Progs_{A'} \Rightarrow \mathcal{L}(p) = l) \textbf{ for } OS), \\
OS \textbf{ says } Pgm \in Progs_{A'}, \\
A' \textbf{ says } (Pgm \in Progs_{A'} \Rightarrow \mathcal{L}(Pgm) = l_{Pgm}), \\
\vdash_{\text{CAL}} \\
(9.40)[\underline{\mathrm{f}} := F]
\end{array}$$

This guard sequent specifies derivation trees that constitute support for authorizing a read access to a file F by a program Pgm. In an actual guard implementation, the credentials to convey "A **says** ..." and "A' **says** ..." would not accompany the request but instead the guard would fetch these from classification authorities A and A' respectively when labels for F and Pgm are needed to authorize a specific request.

Exercises for Chapter 9

9.1 A proposal has been made to replace Conservative Approximation for Worldviews clause (ii) by

$$\mathcal{C}_1,\ \mathcal{C}_2,\ \ldots,\ \mathcal{C}_N \vdash_{\text{CAL}} \mathcal{C}$$

Which, if any, CAL inference rules become unsound if this replacement is made?

9.2 Conservative Approximation for Worldviews clause (iii) on page 217 incorporates all beliefs from $\omega(P')$ into $\omega(P)$ when "P' **speaksfor** P" $\in \omega(P)$. Discuss advantages and disadvantages of the following alternative for that clause.

(iii) Add beliefs from $\omega(P')$ into $\omega(P)$ when "P' **speaksfor** P" $\in \omega(P')$.

9.3 Explain why Conservative Approximation for Worldviews on page 216 ensures that "P_i **speaksfor** P_k" $\in \omega(P)$ holds if both "P_i **speaksfor** P_j" $\in \omega(P)$ and "P_j **speaksfor** P_k" $\in \omega(P)$ hold.

9.4 A proposal has been made to replace Conservative Approximation for Worldviews clause (iv) by

(iv) Add all formulas $\mathcal{C}'$ where $\mathcal{C}[x := \tau] \Rightarrow \mathcal{C}'$ for any term τ, $\mathcal{C}[x := \tau] \in \omega(P')$ and "P' **speaks** $x{:}\mathcal{C}$ **for** P" $\in cl_{\text{CAL}}(P, B)$.

Does this alter the contents of the worldview for any principal?

9.5 Prove that if "P **says** $\mathcal{C}$" $\in \omega(P)$ and "P **says** $(\mathcal{C} \Rightarrow \mathcal{C}')$" $\in \omega(P)$ then "P **says** $\mathcal{C}'$" $\in \omega(P)$ will hold.

9.6 Exhibit a CAL sequent $\mathcal{C}_1,\ \mathcal{C}_2\ \ldots,\ \mathcal{C}_N \vdash_{\text{CAL}} \mathcal{C}$ for which conclusion $\mathcal{C}$ can be derived from assumptions $\mathcal{C}_1$, ..., $\mathcal{C}_N$ only by using REST-NARROW. (Doing so establishes the necessity of having the rule.)

9.7 If $\omega(P') \subseteq \omega(P)$ holds will P' **speaksfor** P necessarily be a theorem of CAL? Explain.

9.8 To prove that a derived inference rule

$$\text{R:}\frac{\mathcal{C}_1,\ \mathcal{C}_2,\ ...,\ \mathcal{C}_n}{\mathcal{C}}$$

of CAL is a sound, it suffices to exhibit a derivation tree schema

$$\mathcal{T}_{\text{R}}(\underline{\mathsf{C}}_1,\ \underline{\mathsf{C}}_2,\ ...,\ \underline{\mathsf{C}}_n,\ \underline{\mathsf{C}})$$

where meta-variables $\underline{\mathsf{C}}_1$, $\underline{\mathsf{C}}_2$, ..., $\underline{\mathsf{C}}_n$, $\underline{\mathsf{C}}$ denote CAL formulas and the resulting derivation trees it produces (i) does not contain instances of inference rule R, (ii) has $\underline{\mathsf{C}}$ as its conclusion, and (iii) has $\underline{\mathsf{C}}_1$, $\underline{\mathsf{C}}_2$, ..., $\underline{\mathsf{C}}_n$ as its only uncanceled assumptions. For example, soundness of SAYS-IMP-MP follows from derivation tree (9.15). Use this approach to prove that the other inference rules in Figure 9.5 are sound derived inference rules of CAL.

(a) SAYS-AND-I

(b) SAYS-OR-I

(c) SAYS-IMP-I

(d) SAYS-AND-E

(e) SAYS-OR-E

9.9 Consider a computer network where some source process P_0 sends a message m to an intermediary P_1, which in turn forwards m to successor P_2, and so on, until the message reaches its destination P_N. The belief that P_N holds after receiving the forwarded message might be summarized using CAL as

$$P_N \textbf{ says } (P_{N-1} \textbf{ says } (\ \dots\ (P_0 \textbf{ says } m))). \tag{9.46}$$

What additional CAL formulas are plausible uncanceled assumptions to use in a derivation tree for P_0 **says** m from (9.46).

9.10 Is there a goal formula $\mathcal{G}$ that authorizes an operation under some set of credentials but prohibits that same operation if a superset of those credentials is submitted to the guard? Justify your answer.

9.11 The following has been proposed to replace definition (9.20) on page 222 for worldview $\omega(P.\eta)$ of a subprinicipal $P.\eta$:

$$\omega(P.\eta) \;\; = \;\; cl_{\mathrm{CAL}}(P.\eta,\ Init_{P.\eta} \cup Init_P).$$

Would SUBPRIN be sound if this new definition were used? Explain why or why not.

9.12 Prove (9.21)

$$cl_{\mathrm{CAL}}(P_G^{\wedge}\,,\ \bigcap_{P\in G}\omega(P)) \;\; = \;\; \bigcap_{P\in G}\omega(P)$$

mentioned in the discussion of conjunctive group principals.

9.13 Let $G = \{P_1, ..., P_m\}$ be a finite set of principals.

(a) Show that $P_G^{\wedge}$ **speaksfor** $P_G^{\vee}$ is valid by proving $\omega(P_G^{\wedge}) \subseteq \omega(P_G^{\vee})$.

(b) Determine whether

$$\wedge\text{-DELEG-}\vee\text{:}\frac{}{P_G^{\wedge} \textbf{ speaksfor } P_G^{\vee}}$$

is a sound derived inference rule of CAL, and justify your determination. (Exercise 9.8 discusses proof obligations for demonstrating soundness of a CAL derived inference rule.)

9.14 Let $G = \{P\}$ be a singleton set of principals.

(a) Show that

$$P_G^{\wedge} \textbf{ speaksfor } P_G^{\vee} \qquad \text{and} \qquad P_G^{\vee} \textbf{ speaksfor } P_G^{\wedge}$$

both are valid by proving $\omega(P_G^{\wedge}) = \omega(P_G^{\vee})$.

(b) Determine whether

$$\wedge\text{-=-}\vee\text{:}\frac{G = \{P\}}{P_G^{\wedge} \textbf{ speaksfor } P_G^{\vee}} \qquad \text{and} \qquad \vee\text{-=-}\wedge\text{:}\frac{G = \{P\}}{P_G^{\vee} \textbf{ speaksfor } P_G^{\wedge}}$$

are sound derived inference rules of CAL, and justify your determination. (Exercise 9.8 discusses proof obligations for demonstrating soundness of a CAL derived inference rule.)

9.15 Some logics designed for credentials-based authorization provide special-purpose compound principals for specifying that fine-grained accountability can be specified. Give CAL formulas that might be used to replace each of the following constructions. Justify each answer by showing that appropriate distinctions in accountability are preserved (e.g., despite delegations).

(a) "(P **for** R) **says** $\mathcal{C}$" which is intended to signify that $\mathcal{C}$ should be attributed to principal P serving in role R (as distinct, say, from P serving in other roles or other principals serving in role R).

(b) "(P **quoting** R) **says** $\mathcal{C}$", which is intended to signify that P holds R accountable for $\mathcal{C}$

(c) "(P **as** R) **says** $\mathcal{C}$", which is intended to signify that P is acting under a name that makes it accountable for some different set of beliefs than usual.

9.16 Consider a set $G = \{P_1, \ldots, P_N\}$ of principals, where each knows private key k (so k is not all that private) corresponding to some public key K. We might contemplate defining a new kind of group principal, the *cryptographic* group principal P_G^K. Worldview $\omega(P_G^K)$ contains only those beliefs $\mathcal{C}$ for which a k-signed bit string $\mathcal{S}_k$("K **says** $\mathcal{C}$") has been created by some principal.

(a) Give a formal definition for worldview $\omega(P_G^K)$.

(b) Does CAL remain sound if cryptographic group principals like P_G^K are added. Explain why or why not.

9.17 With an (m, n)-*threshold digital signature scheme*, random *shares* s_k^1, s_k^2, ..., s_k^n are generated from a private key k. Knowledge of share s_k^i is necessary and sufficient for constructing s_k^i*-partially-signed* bit string $\mathcal{S}_k^i(b)$ for any given bit string b. Moreover, not only can a k-signed bit string $\mathcal{S}_k(b)$ be computed in the usual way, but it can be computed from any set comprising exactly m

distinct partially-signed bit strings for b—a publicly-known function $TDSS_{(m,n)}$ is invoked with that set of partially-signed bit strings as arguments:

$$\mathcal{S}_k(b) \quad = \quad TDSS_{(m,n)}(\mathcal{S}_k^1(b), \mathcal{S}_k^2(b), ..., \mathcal{S}_k^m(b))$$

Consider a set $G = \{P_1, \ldots, P_N\}$ of principals. An *N-replicated group principal* $P_G^{(N,N)}$ is characterized by

$$(\mathcal{C} \in \omega(P_1) \ \wedge \ \ldots \ \wedge \ \mathcal{C} \in \omega(P_N)) \ \Rightarrow \ \mathcal{C} \in \omega(P_G^{(N,N)})$$

so, by definition, beliefs in $P_G^{(N,N)}$ are necessarily held by all N constituents in G.

(a) Give a formal definition for a worldview $\omega(P_G^{(N,N)})$ being sure that all inference rules of CAL remain sound.

(b) Suppose that there is a well known public key K_G associated with G, that corresponding private key k_G is known to no principal, but that principal P_i in G is the sole principal that knows share $s_{k_G}^i$ of k_G. Propose a protocol for issuing a credential that conveys K_G **says** $\mathcal{C}$ for any belief $\mathcal{C}$ where $\mathcal{C} \in \omega(P_i)$ holds for all principals $P_i \in G$.

(c) Give an argument that K_G **speaksfor** $P_G^{(N,N)}$ is sound.

(d) Do CAL inference rules $\wedge$-GROUP-SAYS-I, $\wedge$-GROUP-DELEG, and $\wedge$-GROUP-SAYS-E become unsound if conjunctive group principal $P_G^{\wedge}$ is replaced by replicated group principal $P_G^{(N,N)}$.

9.18 Defining characteristic (9.30) of beliefs in worldview $\omega(\mathcal{H}(b \rhd rep(\mathcal{C})))$ incorporates beliefs $\mathcal{C}'$ whose representation do not appear in $b \rhd rep(\mathcal{C})$ nor are derived from the belief $\mathcal{C}$ that does appear. Worldview $\omega(\mathcal{H}(b \rhd rep(\mathcal{C})))$ therefore contains what might be termed *spurious beliefs.* We might consider substituting

$$Init_{\mathcal{H}(b \rhd rep(\mathcal{C}))} \quad = \quad \{\mathcal{C}\}$$

in the definition of $\omega(\mathcal{H}(b \rhd rep(\mathcal{C})))$ as an alternative for avoiding spurious beliefs. If it works, what advantages does the alternative offer over the scheme given in §9.6.2? Does the alternative work?

9.19 One approach to kernel support for credential integrity is outlined in §9.6.3. We can use CAL to give a formal account of this approach by considering the operating system to be a principal, OS.

(a) What properties should $Init_{OS}$ satisfy with regard to beliefs that processes hold and/or beliefs that are being stored in WV?

(b) Suppose the representation of some belief $\mathcal{C}$ (say) appears in the set returned to P after invoking *QueryCred* in Figure 9.6. Explain why the characterization of that returned belief ought to be "OS **says** $\mathcal{C}$" rather than simply "$\mathcal{C}$".

(c) What CAL assumption is both defensible and suffices for P to construct a derivation tree having conclusion P **says** $\mathcal{C}$ from "OS **says** $\mathcal{C}$" that *QueryCred* returns.

9.20 An authorization policy is not given for the operations in Figure 9.6. Under what assumptions is it feasible to enforce an authorization policy along the following lines:

(i) every belief is either public or secret,

(ii) *QueryCred* returns public beliefs but not secret beliefs, and

(iii) P has complete control over whether a belief having form "P **says** $\mathcal{C}$" can become known to other principals.

9.21 Many file systems associate an access control list with each file. When separate privileges `r` (read), `w` (write), and `x` (execute) are associated with the different operations, then the access control list for a file F defines sets $ACL_{\mathtt{r}}(F)$, $ACL_{\mathtt{w}}(F)$, $ACL_{\mathtt{x}}(F)$ of principals authorized to perform the designated operations. For a file system like the one sketched in §9.7 having goal formula (9.35), we could interpret $P \in ACL_{\Theta}(F)$ as

$$\mathtt{FileSys}\ \textbf{says}\ (P\ \textbf{speaks}\ \Theta(F)\ \textbf{for}\ \mathtt{FileSys})$$

An alternative design is to use

$$\mathtt{FileSys}\ \textbf{says}\ P \in ACL_{\Theta}(F)$$

as the goal formula for P performing operation $\Theta(F)$. Discuss the advantages and disadvantages of this alternative goal formula.

9.22 Recall (from chapter 7) that a capability (i) conveys a pair $\langle O, Privs\rangle$, where O is an object and $Privs$ is a set of privileges, and (ii) is represented in a way that cannot be counterfeited or corrupted.

(a) Discuss the similarities and differences of capabilities and credentials.

(b) Describe an implementation for capabilities in terms of credentials and goal formulas. Assume that a guard exists for every operation Θ that requires holding a capability that authorizes Θ.

9.23 According to Figure 9.9, a separate symmetric key is required for each client when MAC-protected messages are used to convey responses from an authority. What additional assumptions about communications channels would allow a single symmetric key to be used for messages being sent to a set of clients?

9.24 Why aren't nonces needed in responses from authorities when each of the following mechanisms is used for communication between an authority and client?

(a) a system calls

(b) system-provided IPC channels

(c) MAC-protected messages

9.25 Suppose a classification authority A issues credentials that convey

$$A \textbf{ says } (F \in Obj_A \wedge \mathcal{L}(F) = l_F)$$

instead of

$$A \textbf{ says } (F \in Obj_A \Rightarrow \mathcal{L}(F) = l_F).$$

Should you endorse changing to this stronger credential? What benefits and/or risks does this stronger credential bring?

9.26 A principal P that is not authorized to read a file F might nevertheless be authorized to read a file $\texttt{San}(F)$ that is generated by some *sanitization* program San. For instance, sanitization to support a read operation under multi-level security might delete or modify sensitive contents to ensure $\mathcal{L}(\texttt{San}(F)) \preceq \mathcal{L}(P)$ holds even if $\mathcal{L}(F) \preceq \mathcal{L}(P)$ does not.

(a) Revise goal formula template (9.40) and the rest of that account to accommodate sanitization that is authorized by the classification authority that assigns labels to a given file.

(b) Revise goal formula template (9.40) and the rest of that account to accommodate sanitization that is authorized by $own(\texttt{f})$.

9.27 Sketch a credentials-based authorization scheme for each of the following. The sketch should include an appropriate guard sequent, an explanation of what state the guard maintains, pseudo-code for state updates, and a description of how the guard obtains credentials needed to authorize an access request.

(a) Only the owner may perform operations on resource R.

(b) Each access request by a principal must be explicitly endorsed by that principal's manager.

(c) Multi-level Confidentiality is enforced, except trusted subjects may perform "write down".

(d) An individual is not allowed to read the records for two or more companies that compete with each other.

(e) Operations on *Obj* are performed in a pre-specified order. In particular, $\Theta_{i+1}(Obj)$ is performed by a principal only after $\Theta_i(Obj)$ has been performed by some principal.

(f) Access is permitted only by those users who are over 21 years old.

(g) After a request by P for `open`(F) has been performed, P may perform at most 10 operations on F before another request for `open`(F) must be performed by P.

(h) The request to read F must be from some authorized server that is performing an operation for some user U that it authorized to undertake operation $\Theta(F)$.

(i) All system administrators are allowed to add or delete new users to the system.

(j) The company has a set *Divs* of divisions and each division D has a set $Mngrs_D$ of managers. Every request must be approved by some manager from each division.

(k) Requests must have originated by a program on a computer executing `Unix2.0`.

Notes and Reading

Logical inference for making authorization decisions—a hallmark of credentials-based authorization—goes a step beyond using authentication protocols to attribute access requests. The approach derives from research [3, 21, 33] from the late 1980's into building secure distributed systems at Digital Equipment Corporation's System Research Center (DEC SRC) in Palo Alto CA and a parallel effort to produce an architecture specification, *The Digital Distributed System Security Architecture* [13], by Digital engineers based on the East Coast. In both, a rich language for defining principals allowed access control lists to be used for authorizing requests that originate remotely. Rather than equating principals with users, principals were defined in terms of components (users, hardware, software, and their aggregations) that together would be accountable for a given remote access. A calculus—which included **says**, **speaksfor**, and various operators for constructing compound principals—characterized ways that statements attributed to various principals might be combined to derive an access request attributed to some (possibly different) principal appearing on the relevant access control list. The syntax of goal formulas in this early embodiment of credentials-based authorization was constrained so that the operating system could be programmed to decide automatically whether an accompanying set of credentials sufficed to authorize a given access request. Earlier work [7] at DEC SRC used deductions and formulas involving "say", "knows", and "authenticates" (a form of **speaksfor** for keys) to describe a distributed authentication service; that work seems to be the first use of the **says**/**speaksfor** metaphor in connection with security.

PolicyMaker [8], which came next and was developed at Bell Laboratories, avoided sacrificing expressiveness for decidability. Policies, credentials, and trust relationships were specified as imperative programs in a safe language, and a generic compliance checker interpreted these programs to determine whether a policy is satisfied by given credentials and trust assumptions. PolicyMaker's intended users were presumed to be more comfortable writing imperative programs than writing logical formulas. But determining whether some program in an imperative language satisfies a property of interest is difficult—for people as well as for machines. So our assurance is likely to be lower for a security policy that is specified as an imperative program, and our ability to anticipate its consequences impaired. Contrast this with security policies specified using logical formulas, where inference rules support reasoning and derivations for consequences can be checked mechanically.

Prolog, Datalog, and related languages for writing *logic programs* offer a compromise between expressiveness and decidability. Here, a security policy is specified declaratively as a collection of rewrite rules. Derivation of a goal formula from such rules is decidable, and having such a derivation constitutes a justification for authorizing an access. Early efforts to explore this approach include (chronologically): an authorization policy simulator [25], FAM/CAM [17], ASL [16], Delegation Logic [23], SD3 [18], Binder [10], the RT family of logics [24], Cassandra [6], Soutei [27], and SecPAL [5].

With *proof carrying authentication* (PCA) [4], virtually no expressiveness limitations are imposed on goal formulas but each request must be accompanied by a derivation tree for the appropriate goal formula. The programmer of a client presumably provides code to generate the derivation tree for supporting that client's accesses. Checking a derivation tree is decidable, so an operating system can determine automatically whether the accompanying derivation tree justifies allowing an access request to proceed.

Garg and Pfenning [12] give a constructive logic with first-order quantification over principals (but no **speaksfor** primitive) and give proofs of non-interference and other meta-properties for that logic. This paper could be the first publication to argue that authorization logics ought to be constructive on the grounds that all of the evidence justifying an access decision will then necessarily be incorporated into the derivation of a goal formula from credentials. The choice between classical and constructive is just one dimension in the design space for authorization logics, though. Abadi [2] explores another by deriving consequences of incorporating various combinations of seemingly reasonable axioms for **says** into classical logics and into constructive logics.

CAL is a successor to Nexus Authorization Logic (NAL) [28], which was developed as part of the Nexus [29] operating system built at Cornell. Application needs led the Nexus group to investigate techniques (discussed in §9.8) for accommodating changes in beliefs and for supporting revocation of credentials; DeTreville [10] was earlier down this path in connection with certificate revocation for Binder. And concerns about the performance of Nexus led to the development (sketched in §9.6.3) of kernel caches for credentials and for derivation trees. A Nexus document-management suite [28, 32] that supports multi-level security policies along with various forms of document-use policies is the source of the classification authority formalization in §9.9.

The original plan for Nexus was to adapt prior work rather than developing a new authorization logic. Early papers [3, 4] about authorization logics suggest an abbreviation

$$P\ \textbf{controls}\ p\colon\quad (P\ \textbf{says}\ p) \Rightarrow p$$

for signifying when P is considered a trusted source about the truth of a predicate p. This approach, however, imposes no limits on the propagation of inconsistencies and bogus beliefs. An alternative approach is to require that predicates about the state be represented solely through beliefs that principals hold. Delegation then enables beliefs about state predicates one principal holds to become accessible to others, and non-interference ensures inconsistent or bogus beliefs at one principal P are contained—contamination in P's worldview can impact worldviews of only those principals that (directly or indirectly) delegate to P.[18]

NAL adopted this second alternative, taking as its starting point CDD and the constructive first-order predicate logic in van Dalen [31]. Because CDD is agnostic about forms of compound principals, NAL was able to incorporate compound principals that were well-matched to what Nexus required. NAL subprincipals are inspired by named roles in Alpaca [22].[19] Groups in NAL are specified intensionally by giving a predicate that all members satisfy; this is a special case of *dynamic threshold structures* from Delegation Logic [23].

CAL simplifies NAL. First, NAL (being a CDD derivative) includes second-order quantification, which allows **speaksfor** to be a derived operator:

$$P\ \textbf{speaksfor}\ Q\colon\quad (\forall \mathcal{C}\colon\quad P\ \textbf{says}\ \mathcal{C} \quad\Rightarrow\quad Q\ \textbf{says}\ \mathcal{C})$$

CAL, for pedagogical reasons, has only first-order quantifiers, which are restricted to appearing within predicate logic formulas. Therefore, **speaksfor** is a primitive in CAL rather than a derived operator; CAL inference rules for unrestricted and restricted delegation are derived inference rules in NAL.

Principals are the other significant point of difference between CAL and NAL. CAL gives formal accounts for keys and hashes as full-fledged principals; NAL treated these informally. In addition, CAL replaces NAL's intensionally defined groups with conjunctive and disjunctive group principals. Conjunctive groups are discussed and axiomatized in Lampson et al. [21]. CAL's disjunctive group principals axiomatize what Syverson and Stubblebine [30] call a *collective group*. CAL cannot support what they call an *or-group* [30] (a group G where G **says** $\mathcal{C}$ only if P **says** $\mathcal{C}$ for some member P of group G) because an or-group's worldview is not necessarily closed under logical deduction and, thus, CAL inference rule SAYS-IMP-E would not be sound.

Modal logics are finding increased application in connection with software systems. Besides credentials-based authorization logics, modal logics for reasoning about event ordering (so called *temporal logics*) are widely used for reasoning

[18]The need for decoupling the beliefs of different principals had first been noted by Abadi et al. [3, §3.2], and the description of CDD [1] made explicit the connection between non-interference (by then an already well established term for information-flow policies) and information-flow type systems. Garg and Pfenning [12] were the first to formalize "non-interference" for an authorization logic.

[19]Prior proposals (e.g., Taos [33]) had restricted the qualifier η used in defining a subprincipal $A.\eta$ to being a fixed string, which meant that only static roles could be supported.

about concurrent programs, and modal logics for reasoning about beliefs held by individuals and groups (so called *epistemic logics*[20]) have been used for reasoning about coordination in distributed systems and in AI systems that perform reasoning. Hughes and Cresswell [15] is an excellent introductory textbook on modal logics.

Kripke structures [20], which were developed expressly for giving formal semantics to modal logics, are employed by Abadi et al. [3] for the authorization logic developed at DEC SRC. We adopted an alternative kind of structure for CAL's semantics, inspired by PCA [4] which interprets formulas with respect to "worldviews" that are sets of formulas closed under logical implication. Sets of formulas were being used in the early 1980's by Konolige [19] for belief logics intended to support AI planning systems, preceded by Eberle [11] as well as Moore and Hendrix [26]. Sets of formulas also are used for BAN logic [9] (an epistemic logic named after it's authors $\underline{\text{B}}$urrows, $\underline{\text{A}}$badi, and $\underline{\text{N}}$eedham that was designed for proving properties of cryptographic authentication protocols). Hirsch and Clarkson [14] show how a semantics based on the worldviews is related to one based on Kripke structures and also give a worldviews semantics for a logic derived from NAL; our formalization of CAL satisfaction relation $\models_{\text{CAL}}$ is based on this work.

Bibliography

[1] Martín Abadi. Access control in a core calculus of dependency. In *Proceedings of the Eleventh ACM SIGPLAN International Conference on Functional Programming*, ICFP '06, pages 263–273, New York, NY, USA, 2006. ACM.

[2] Martín Abadi. Variations in access control logic. In Ron Meyden and Leendert Torre, editors, *Deontic Logic in Computer Science*, volume 5076 of *Lecture Notes in Computer Science*, pages 96–109. Springer Berlin Heidelberg, July 2008.

[3] Martín Abadi, Michael Burrows, Butler W. Lampson, and Gordon D. Plotkin. A calculus for access control in distributed systems. In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 1991.

[4] Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, CCS '99, pages 52–62, New York, NY, USA, 1999. ACM.

[5] Moritz Becker, Cedric Fournet, and Andrew Gordon. Design and semantics of a decentralized authorization language. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium*, CSF '07. IEEE Computer Society, July 2007.

[20]The term *epistemic* means "relating to knowledge or beliefs".

[6] Moritz Y. Becker and Peter Sewell. Cassandra: Flexible trust management, applied to electronic health records. In *Proceedings of the 17th IEEE Workshop on Computer Security Foundations*, CSFW '04, pages 139–154. IEEE Computer Society Press, June 2004.

[7] Andrew D. Birrell, Butler W. Lampson, Roger K. Needham, and Michael D. Schroeder. A global authentication service without global trust. In *Proceedings of 1986 IEEE Symposium on Security and Privacy*, pages 223–230. IEEE Computer Society Press, 1986.

[8] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, 1996.

[9] Michael Burrows, Martín Abadi, and Roger Needham. Authentication: A practical study in belief and action. In *Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning about Knowledge*, TARK '88, pages 325–342, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc.

[10] J. DeTreville. Binder, a logic-based security language. In *Proceedings of 2002 IEEE Symposium on Security and Privacy*, pages 105–113. IEEE Computer Society Press, 2002.

[11] Rolf A. Eberle. A logic of believing, knowing, and inferring. *Synthese*, 26(3-4):356–382, 1974.

[12] Deepak Garg and Frank Pfenning. Non-interference in constructive authorization logic. In *Proceedings of the 12th IEEE Workshop on Computer Security Foundations Workshop*, CSFW '09, pages 283–296. IEEE Computer Society Press, July 2006.

[13] Morrie Gasser, Andy Goldstein, Charlie Kaufman, and Butler Lampson. The digital distributed system security architecture. In *Proceedings of 12th National Computer Security Conference*, pages 305–319. National Institute of Standards and Technology, National Computer Security Center, October 1989.

[14] Andrew K. Hirsch and Michael R. Clarkson. Belief semantics of authorization logic. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pages 561–572, New York, NY, USA, 2013. ACM.

[15] G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic*. Routledge, 1996.

[16] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *Proceedings of 1997 IEEE Symposium on Security and Privacy*, pages 31–42. IEEE Computer Society Press, 1997.

[17] Sushil Jajodia, Pierangela Samarati, V. S. Subrahmanian, and Eliza Bertino. A unified framework for enforcing multiple access control policies. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, SIGMOD '97, pages 474–485. ACM, 1997.

[18] T. Jim. SD3: A trust management system with certified evaluation. In *Proceedings of 2001 IEEE Symposium on Security and Privacy*, pages 106–115. IEEE Computer Society Press, 2001.

[19] Kurt Konolige. A deductive model of belief. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 1, pages 377–388, 1983.

[20] Saul Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[21] Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. In *Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles*, SOSP '91, pages 165–182, New York, NY, USA, 1991. ACM.

[22] Chris Lesniewski-Laas, Bryan Ford, Jacob Strauss, Robert Morris, and M. Frans Kaashoek. Alpaca: Extensible authorization for distributed services. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 432–444, New York, NY, USA, 2007. ACM.

[23] Ninghui Li, Joan Feigenbaum, and Benjamin N. Grosof. A logic-based knowledge representation for authorization with delegation. In *Proceedings of the 12th IEEE Workshop on Computer Security Foundations*, CSFW '99. IEEE Computer Society Press, June 1999.

[24] Ninghui Li and John C. Mitchell. Design of a role-based trust management framework. In *Proceedings of 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, 2002.

[25] J.D. Moffett and M.S. Sloman. The source of authority for commercial access control. *Computer*, 21(2):59–69, 1988.

[26] Robert C. Moore and Gary G. Hendrix. Computational models of beliefs and the semantics of belief-sentences. Technical Report 187, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, June 1979.

[27] Andrew Pimlott and Oleg Kselyov. Soutei, a logic-based trust management system, system description. In M. Hagiya and P. Wadler, editors, *Proceedings 8th International Symposium on Functional and Logic Programming (FLOPS 2006)*, volume 3945 of *Lecture Notes in Computer Science*, pages 130–145, April 2006.

[28] Fred B. Schneider, Kevin Walsh, and Emin Gün Sirer. Nexus authorization logic (NAL): Design rationale and applications. *ACM Transactions on Information Systems Security*, 14(1):8:1–8:28, June 2011.

[29] Alan Shieh, Dan Williams, Emin Gün Sirer, and Fred B. Schneider. Nexus: A new operating system for trustworthy computing. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles*, SOSP '05, pages 1–9, New York, NY, USA, 2005. ACM.

[30] Paul F. Syverson and Stuart G. Stubblebine. Group principals and the formalization of anonymity. In *In World Congress on Formal Methods*, pages 814–833. Springer-Verlag, September 1999.

[31] D. van Dalen. *Logic and Structure*. Universitext (1979). Springer, 2004.

[32] Kevin A. Walsh. *Authorization and Trust in Software Systems*. PhD thesis, Cornell University, January 2012.

[33] Edward Wobber, Martín Abadi, Michael Burrows, and Butler Lampson. Authentication in the Taos operating system. In *Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles*, SOSP '93, pages 256–269, New York, NY, USA, 1993. ACM.