A Formal Semantics for SPKI

Jon Howell[*]
David Kotz

Technical Report TR 2000-363
Department of Computer Science
Dartmouth College
Hanover, NH 03755-3510
{jonh,dfk}@cs.dartmouth.edu

March 14, 2000

Abstract

We extend the logic and semantics of authorization due to Abadi, Lampson, *et al.* to support restricted delegation. Our formal model provides a simple interpretation for the variety of constructs in the Simple Public Key Infrastructure (SPKI), and lends intuition about possible extensions. We discuss both extensions that our semantics supports and extensions that it cautions against.

1 Introduction

This paper provides a formal semantics for the Simple Public Key Infrastructure (SPKI), an Internet Experimental Protocol [EFL+99]. The current (2.0) version of SPKI is a merger of SPKI 1.0 and the Simple Distributed Security Infrastructure (SDSI) 1.0.

SPKI is an elegant practical system that addresses the problem of ensuring that a user is *authorized* to perform an action, not just the problem of identifying the user. This focus allows for much more flexible sharing of resources through delegation; in contrast, systems based on authentication with a conventional public-key infrastructure (PKI) plus authorization with conventional ACLs limit the available modes of resource sharing. SPKI does incorporate a notion of authentication as well: its linked local namespaces bind keys to names. This notion of authentication is more general than conventional hierarchical PKI naming, allowing it to escape the "trusted-root" problem.

Unfortunately, SPKI is not founded on a formal semantics that can provide intuition for what it does, what it promises, what it assumes, and how it may or may not be safely extended.

Abadi, Lampson, and others defined an authorization system called the Calculus for Access Control [ABLP93, LABW92]. This system provides delegation without restrictions. A user can encode restrictions by delegating control over "self as *role*" to another user, and adding the principal "self as role" to the ACL of the resource to be shared. The system is based on a formal semantics that explains how delegations interact with various combination operators for principals. Our formalism for SPKI is based on the semantics of the Calculus for Access Control, extended to support restricted delegation and SPKI names.

Our formal treatment of SPKI is attractive for two reasons:

First, it supplies intuition for what SPKI operations mean. The proliferation of concrete concepts in SPKI can be understood as applications of just three abstractions: *principal*, *statement*, and *name*.

[*]Supported by a research grant from the USENIX Association.

Second, the formalism gives us guidance in extending SPKI. We give an example of a dangerous extension that the formalism advises against, and we give examples of extensions that the formalism supports and that we use in our concrete system implementation.

We begin in Section 2 by discussing related formalisms. We then provide a quick overview of modal logic and possible-worlds semantics in Section 3, followed by reviews of the original Calculus for Access Control and SPKI in Sections 4 and 5. In Section 6, we describe our extension to the Calculus for Access Control and our extensions to its semantics to support restricted delegation. Section 7 further extends the formalism to support SPKI linked local names, and discusses the shortcomings of Abadi's semantics for SPKI names. We model SPKI tags in Section 8. In Section 9, we use the work from the prior two sections to construct a formal scaffolding under SPKI, and we discuss some of the interesting ramifications of our formalism. We summarize our contributions in Section 10. Proofs appear in the appendix.

2 Related work

Abadi provides a semantics for SPKI names in [Aba98], but its definition shares a flaw with that used for roles in [ABLP93]. We discuss Abadi's name semantics in Section 7.3.

Halpern and van der Meyden supply an alternate semantics for SPKI names in [HvdM99], but it only encompasses the containment relation among names, and does not treat names as principals. As a result, it cannot relate names to compound principals nor relate names to other principals that are only connected by a restricted delegation.

Aura supplies a semantics for SPKI restricted delegation in [Aur98], but it is unsatisfying in that it essentially says what the reduction procedure says: a delegation is in place if there is a chain of delegation certificates and principals. It does not lend intuition about what the delegations mean. In contrast, our semantics connects restricted delegation to the logic of belief, a formal model that describes what a principal means when it delegates authority.

3 Review: the logic of belief

> *The Sicilian smiled and stared at the wine goblets. "Now a great fool," he began, "would place the wine in his own goblet, because he would know that only another great fool would reach first for what he was given. I am clearly not a great fool, so I will clearly not reach for your wine."*
>
> *"That's your final choice?"*
>
> *"No. Because you knew I was not a great fool, so you would know that I would never fall for such a trick. You would count on it. So I will clearly not reach for mine either."* [Gol73, p. 157]

The Sicilian's great effort went into reasoning about the beliefs of his opponent, including his opponent's beliefs about his own beliefs, and so on. His watertight reasoning is an example of modal logic, the logic of belief. One way to reason about permissions and sharing is to reason about who believes what. We call participants in a distributed system *agents*, and the symbols that represent agents in logical expressions *principals*. Principals can also represent sets of agents, or one agent quoting another; these are called *compound principals*, and we discuss them in Section 3.1. If Alice believes everything Bob believes (that is, Alice trusts Bob in every matter), then if Bob believes it is good to read a given file, Alice must believe the same. In this section, we develop a model for reasoning about logic in the presence of belief.

We begin with propositional logic. Assume there is a set of primitive (uninterpreted, independent) statements Σ.[1] For our purposes of access control, we consider primitive statements such as "it is good to write to file X." This interpretation turns an imperative command into a declarative proposition. The primitive statements may be connected with *and* ($\wedge$) and *not* ($\neg$) to form arbitrary formulas. The *or* ($\vee$)

[1] Figure 3 provides a table of sets and variable notation used in this paper.

and *implies* ($\supset$) operators are abbreviations for longer formulas made of $\wedge$ and $\neg$.

Next we introduce a *modal* operator **believes**.[2] If σ is a formula and principal A represents agent Alice, $A\,\mathbf{believes}\,\sigma$ is a formula that can be read "Alice believes σ is true." In time, we will introduce multiple **believes** operators, one per principal. For now, we would like to build a *model* that helps us understand which formulas A believes; that is, for which σ do we have $A\,\mathbf{believes}\,\sigma$?

To model this logic, we build a *Kripke structure*. A Kripke structure is a tuple of sets $\mathcal{M} = \langle W, I, J\rangle$. The members of set W represent *possible worlds*. The function I maps a primitive proposition (s) to the set of worlds where it is true, and the function J maps a principal to a relation on worlds in W. Together, I and J determine the truth value of every formula in every world in W; we describe them in more detail shortly.

First, some intuition: A principal A living in world w_0 considers some other set of worlds possible, and if a formula σ is true in each of those other worlds, then A believes the formula. The interesting thing about possible worlds is that the set of worlds A considers possible captures what she does not know: if a statement σ appears in one possible world and $\neg\sigma$ appears in another, then A knows neither σ nor $\neg\sigma$. As far as she is concerned, σ could go either way, because A cannot tell which of the possible worlds she actually is in.

When we write $\mathcal{M}, w_0 \models \sigma$ (pronounced "$\mathcal{M}$ at w_0 *models* σ"), we mean that in model $\mathcal{M}$ at world w_0, the formula σ is true. The mapping I tells us immediately about the truth of primitive propositions at different worlds, but we wish to determine the truth of arbitrary statements σ, including propositional connectives and our modal operators ($\sigma = A\,\mathbf{believes}\,\tau$). We illustrate with an example structure, shown in Figure 1.

The model contains three primitive statements, l, b, and p. The statement l means that our agent Alice (A) is in the produce department of a grocery store. Its negation, $\neg l$, means that Alice is in the meat department (it's a small store). The b primitive means that the store's bananas are yellow, and the p primitive means that the store's pork is fresh.

Recall the three parts of a model, $\langle W, I, J\rangle$. W is the set of possible worlds; in our case, since there are three primitive statements, there are at most eight: $W = \{w_0, w_1, \ldots w_7\}$. I is a relation that defines which primitive statements are true at which worlds. In our example, $I(b) = \{w_0, w_1, w_4, w_5\}$, since the bananas are only yellow in those four worlds. Finally, J is a function that maps principals to relations. Because we have only one principal (Alice), J has only one mapping, written $J(A)$. The relation $J(A)$ is depicted with arrows in the diagram. For example, $\langle w_0, w_1\rangle \in J(A)$; that is, when the actual world is w_0, w_1 is a world Alice considers possible. In our example, it happens that Alice considers two worlds possible from each world.

Assume for a moment that the actual world is in fact w_0: Alice is in the produce department, the bananas are yellow and the pork is fresh. If Alice were omniscient, she would consider only w_0 possible, for that is indeed the state of things. Alice, however, is merely a shopper. She cannot see from the produce department what is going on in the meat department, and thus she cannot tell if the pork is fresh. She must also consider possible world w_1, where the pork is spoiled. She knows for certain her own location, though, so she can ignore worlds $w_4 \cdots w_7$. Because she is in the produce department and can see the bananas, she can also ignore worlds w_2 and w_3 in which the bananas are green.

We have explained the two arrows emanating from world w_0. The other arrows in the diagram, comprising the relation $J(A)$, communicate the same sort of information about any other state of affairs. For example, if the actual world were w_1 (the pork is in fact spoiled), Alice considers just the same worlds w_0 and w_1 possible, and for the same reasons.

Now that you have the intuition behind the Kripke structure, we can formally define when various statements are true. Primitive proposi-

[2] In conventional modal logic, $A\,\mathbf{believes}\,\sigma$ is written $\Box_A\sigma$.

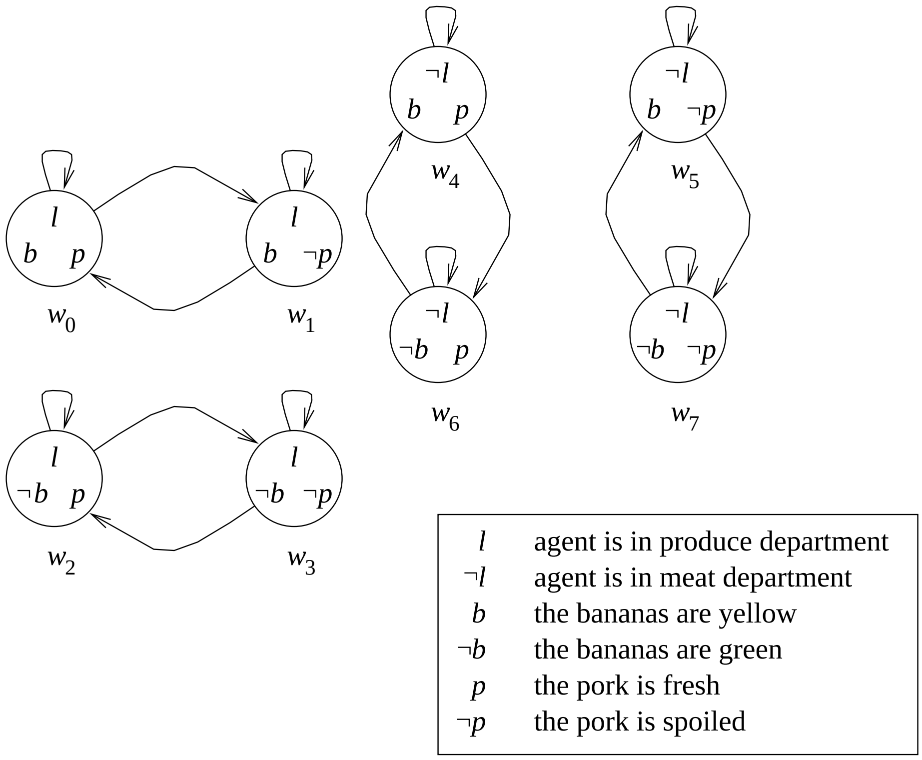


Figure 1: *A model of eight worlds (circles), illustrating the relationship between the accessibility relation (arrows) and the the modal operator (A **believes**).*

tions are easy: the casual definition of I above becomes:

$$\mathcal{M}, w_0 \models s \qquad \text{when } w_0 \in I(s)$$

This definition can be read "Statement s is true at world w_0 in model M when w_0 is in the set $I(s)$."

What about formulas constructed from the propositional connectives $\wedge$ and $\neg$? The truth of some complex formula σ in a world is completely determined by the truth of its primitive propositions, which the model defines by the mapping I. So we can formally define an *extension* function $\mathcal{E}$ to extend the definition of I to arbitrary formulas. $\mathcal{E}$ is defined recursively starting with I, and extends as you would expect for the propositional connectives:

$$\mathcal{E}(s) = I(s)$$
$$\mathcal{E}(\neg\sigma) = W - \mathcal{E}(\sigma)$$
$$\mathcal{E}(\sigma \wedge \tau) = \mathcal{E}(\sigma) \cap \mathcal{E}(\tau)$$

Not surprisingly, $\neg\sigma$ holds in exactly those worlds where σ does not, and $\sigma \wedge \tau$ holds in exactly those worlds where both subformulas hold. Take a look at the example structure and convince yourself that $\mathcal{E}(b \wedge \neg p) = \{w_1, w_5\}$.

We embarked on this journey to discover when Alice believes various statements, so we need to find out when the model supports formulas including our modal belief operator. The natural intuition is that Alice should believe a statement whenever it is true in *every* world Alice considers possible. To recall our example, b is true (the bananas are yellow) in every world Alice considers possible from w_0, so $\mathcal{M}, w_0 \models A\,\mathbf{believes}\,b$. But because Alice considers w_0 and w_1 possible, she considers both p and $\neg p$ possible; and so she can believe neither; hence we have $\neg(A\,\mathbf{believes}\,p)$ and $\neg(A\,\mathbf{believes}\,\neg p)$ at world w_0. (You can think of this situation as representing Alice's "silence" on the matter of p. Even though Alice asserts neither p nor $\neg p$, every formula is assigned a truth value. It is just that both $A\,\mathbf{believes}\,p$ and $A\,\mathbf{believes}\,\neg p$ are false.)

With this intuition, we fill out the definition of $\mathcal{E}$ to mention formulas containing our modal operator $A\,\mathbf{believes}$:

$$\mathcal{E}(A\,\mathbf{believes}\,\sigma) = \{w | J(A)(w) \subseteq \mathcal{E}(\sigma)\}$$

$J(A)(w)$ denotes the set of worlds that A considers possible from w.[3] So when σ is true in every one of these worlds (i.e., $J(A)(w) \subseteq \mathcal{E}(\sigma)$), then A believes σ (i.e. $A\,\mathbf{believes}\,\sigma$).

Of course, security is not very interesting in a world with only one agent. To introduce a second principal, we simply add a new relation $J(B)$ to our model. Now we can reason about what Bob believes ($B\,\mathbf{believes}\,\sigma$), and even about what Alice believes about what Bob believes ($A\,\mathbf{believes}\,B\,\mathbf{believes}\,\sigma$). (In our example, we could certainly discuss Alice's beliefs about her own beliefs, but for our application to access control, that is not very interesting.)

3.1 Compound principals

It is also possible to talk about *compound principals*. Lampson *et al.* define two operators on principals that can be used to make new compound principals. The first is fairly easy to describe: the principal $A \wedge B$ believes only things that both A and B believe. We can define a new possible-worlds relation for the compound principal in terms of the relations for A and B. To do this, we extend the mapping J to a new mapping $\mathcal{R}$ whose domain includes compound principals. Like the definition of $\mathcal{E}$, $\mathcal{R}$ is defined recursively starting with J:

$$\begin{aligned} \mathcal{R}(A) &= J(A) \\ &\qquad \forall \text{ primitive principals } A \\ \mathcal{R}(\mathcal{A} \wedge \mathcal{B}) &= \mathcal{R}(\mathcal{A}) \cup \mathcal{R}(\mathcal{B}) \\ &\qquad \forall \text{ arbitrary principals } \mathcal{A}, \mathcal{B} \end{aligned}$$

And R replaces J's role in the definition of $\mathcal{E}$:

$$\mathcal{E}(\mathcal{A}\,\mathbf{believes}\,\sigma) = \{w | \mathcal{R}(\mathcal{A})(w) \subseteq \mathcal{E}(\sigma)\}$$

[3] Formally, $J(A)(w) = \{w' | \langle w, w' \rangle \in J(A)\}$.

That set union operation is surprising! What's going on? Recall that the more worlds an agent considers possible, the less the agent believes. In our example structure, Alice could not believe p because she considered world w_1 possible, where p was false. Likewise, by taking the union of the relations for principals A and B to get the relation for the compound principal $A \wedge B$, we ensure that the compound principal is at least as ignorant as either of A or B. If A and B disagree on any statement σ, then $A \wedge B$ can see both worlds where σ is true and worlds where it is false, so $A \wedge B$ can have neither belief.

The second operator for forming compound principals is written $B|A$, and pronounced "B *quoting* A." ("Quoting" may seem an odd choice of words when talking about belief; however, when we translate our terminology into that of Lampson *et al.*, it reads more naturally.) This principal captures B's beliefs about A's beliefs: $(B|A)\,\mathbf{believes}\,\sigma$ should be synonymous with $B\,\mathbf{believes}\,(A\,\mathbf{believes}\,\sigma)$.

The relation for the compound principal $B|A$ is the composition of the relations of B and A:

$$\mathcal{R}(B|A) = \mathcal{R}(B) \circ \mathcal{R}(A)$$

What is the intuition for using composition? Suppose we have $\mathcal{M}, w_0 \models B|A\,\mathbf{believes}\,\sigma$: At world w_0, Bob (agent B) believes Alice believes σ. That means that at every world Bob considers possible from w_0 ($\mathcal{R}(B)(w_0)$), Alice believes σ. But Alice only believes σ at those worlds if σ is true at every world Alice can see from those worlds:

$$\bigcup_{w' \in \mathcal{R}(B)(w_0)} \mathcal{R}(A)(w')$$

The composition $\mathcal{R}(B) \circ \mathcal{R}(A)$ relates w_0 to just this set. So $B|A\,\mathbf{believes}\,\sigma$ is true at w_0 exactly when σ is true in every world reachable from w_0 by the composited relation given above as $\mathcal{R}(B|A)$.

3.1.1 The nature of principal relations

Now that we have a formal structure for discussing the beliefs of principals, let us consider what kinds of beliefs are reasonable, and how principals' beliefs should be related to one another's.

Recall our example structure, where in any world, Alice was either ignorant (had no belief) about either the pork or ignorant about the bananas. The first observation is that agents do not need to believe every true thing; statements about which they have neither a positive nor a negative belief represent something the agent is ignorant about.

Furthermore, observe that Alice never believed anything false: in every world, if $A\,\mathbf{believes}\,\sigma$, σ also held in that world. In the parlance of modal logic, we would say Alice's belief is actually *knowledge*: although she does not have all knowledge, everything she believes is in fact true. Why was this the case? Notice that Alice's possible-worlds relation is reflexive: for every world Alice's relation includes an edge pointing back to that world. That is why Alice cannot believe anything false. If σ is not true in a given world, Alice cannot believe σ there, because the definition

$$\begin{aligned} \mathcal{M}, w \models A\,\mathbf{believes}\,\sigma \;\; &\mathbf{iff}\;\; w \in \mathcal{E}(A\,\mathbf{believes}\,\sigma) \\ &\mathbf{iff}\;\; \mathcal{R}(A)(w) \subseteq \mathcal{E}(\sigma) \end{aligned}$$

precludes it.

In modeling access control in the presence of arbitrary principals, however, we should certainly expect that some principals will believe (or at least claim to believe) untrue things. So we make no restriction of reflexivity on the relation that defines a principal's beliefs. Indeed, a principal may have an empty relation at a world: it may consider *no* worlds possible! In that case, at that world, the agent considers every statement true, since every statement is true in all of the zero worlds the agent considers possible. Indeed, the agent believes *false*. The agent's reasoning has become inconsistent; other agents would be wise not to follow this agent's beliefs.

3.1.2 Trust

Agents following one another's beliefs is exactly how we model trust. If Alice establishes that she believes everything Bob believes, then Alice does not have to be present for Bob to read one of her files: if Bob claims that reading the file would be good, Alice must agree, and the file server grants the request. To capture this trust, we observe that Alice is "less ignorant" than Bob: she believes everything Bob believes, and then perhaps more (on which Bob may remain silent). Therefore, from any actual world, Alice should consider possible a subset of the worlds Bob considers possible. When $\mathcal{R}(A) \subseteq \mathcal{R}(B)$, Alice says everything Bob says; if she says even more, it is because she disregards some possible world that leaves Bob's belief ambiguous. You should convince yourself that if Bob believes σ, Alice has to believe the same thing, for she considers possible only a subset of the worlds Bob considers possible.

4 Review: the original Calculus for Access Control

This section contains an introduction to the Calculus for Access Control due to Abadi, Lampson, *et al.*The reader familiar with it may skip to the next section. We have preserved here the names used for formulas in [LABW92]. We explicitly name formulas L1–L3, which are mentioned in passing in [LABW92, p. 273], and formulas A1–A4, which are mentioned in [ABLP93, pp. 712, 714, and 718].

In the preceding section, we introduced an instance of modal logic: propositional logic plus some modal operators capture the possibly ignorant, possibly false beliefs of fallible principals. The semantics we presented, based on Kripke structures, is exactly that used by Abadi to justify the calculus for access control. We introduced the semantics first, though, because conventionally the semantics is the "intuitive model" of the world, and the logic is a system for discovering theorems (statements that are true in every model) and reasoning from premises to conclusions that must appear in the model.

To apply modal logic to access control, Abadi *et al.* rename the operators. First, "believes" is renamed "**says**." This is meant to capture the notion that the logic is *performative*: sometimes when a principal says something, that something becomes true. The act of saying to a fileserver that a file should be modified, given that the fileserver believes you, causes that file to indeed be modified. This renaming makes the quoting operator sound more natural: $B|A$ is Bob quoting Alice. $B|A\,\textbf{says}\,s$ is meant to be a synonym for $B\,\textbf{says}\,A\,\textbf{says}\,s$. "Belief" is still useful intuition, however. The operator is the same; Bob's belief in σ can be inherited by Alice without Alice actually uttering σ.

A logic is a system of axioms and proof rules that let one reason from premises to conclusions: if the premise holds in a model, the conclusion holds as well. The logic of the Calculus is *sound* in that any conclusion proven in the logic holds in the model, but it is not *complete*: there are statements that are true in every model that cannot be proven in the logic. Abadi suggests that in fact the model may be *undecidable*: no logic system is adequate to prove every valid statement of the model.

The logic of access control is the same (up to variations in notation) as the conventional modal logic system K_n. The subscript n indicates that there are multiple modal operators [HC96, FHMV95, p. 51]. We present that system here.

First, we write $\vdash \sigma$ if a statement σ is valid in the logic: either taken as an axiom, or provable as a theorem from other axioms and the proof rules. We prove theorems using the following:

If σ is a tautology of propositional calculus, then $\vdash \sigma$ (Axiom S1)

The axiom lets us pull in the theorems of propositional calculus without explicitly mentioning the axioms and proof rules that produce them.

$$\frac{\vdash \sigma \qquad \vdash \sigma \supset \tau}{\vdash \tau} \qquad \text{(Rule S2)}$$

The proof rule (modus ponens) says that if both σ and the implication $\sigma \supset \tau$ are valid (provable), then τ is provable as well. It lets us prove theorems about formulas that include the modal operators (**says**) by reasoning from premises to conclusions.

We also have the *Distribution Axiom* (known in modal logic as the axiom **K**, from which the name of the system K_n derives):

$$\vdash A\,\textbf{says}\,(\sigma \supset \tau) \supset (A\,\textbf{says}\,\sigma \supset A\,\textbf{says}\,\tau) \qquad \text{(Axiom S3)}$$

Intuitively it means that agents understand and believe all of the consequences of their beliefs. Furthermore, they believe every theorem:

$$\forall A,\ \frac{\vdash \sigma}{\vdash A\,\textbf{says}\,\sigma} \qquad \text{(Rule S4)}$$

That is, agents know all of the theorems of the logic.

There is a subtle but important distinction between implication in the metalogic (the proof rule above) and implication in the logic. The logical symbol $\vdash$ means that the premises on its left prove the conclusions on its right. The proof rule condition $\vdash \sigma$ means that no premises are required to prove σ; that is, σ is a theorem. When that is true, we may conclude $\vdash A\,\textbf{says}\,\sigma$: it is proven that $A\,\textbf{says}\,\sigma$.

In contrast, the corresponding statement in the logic (not the metalogic) does not hold. The statement $\nvdash \sigma \supset A\,\textbf{says}\,\sigma$ is read "it is not provable that σ implies $A\,\textbf{says}\,\sigma$." The premise of the implication is an arbitrary statement σ (unlike the theorem $\vdash \sigma$ in the proof rule); it is not true that principals say every true statement. They say every theorem (those statements true in every world), but not every true statement (those statements true in the actual world from which the statement is being uttered).

4.1 The calculus of principals

The symbol $=$ is an equivalence relation on principals; by $A = B$ we mean that A and B have the same relation and therefore the same beliefs.[4] (Later in the paper we also use $=$ to denote set equality; its use should be clear from context.)

[4] Abadi *et al.* "note that A and B can have the same beliefs without having the same possible worlds relation; however, because principals are identified by their relations in the semantics, we define equality in terms of relations." This is only possible if the model has two distinct worlds in W that belong to all the same I sets; that is, the model has two separate but indistinguishable worlds.

We have presented the logical tools for reasoning about formulas of statements. Recall that we can also combine principals into principal formulas. For example, $A \wedge B$ is the principal that believes (says) only things that A and B agree upon. In the logic, $A \wedge B$ is defined in terms of its relationship to statements:

$$\vdash (A \wedge B)\,\mathbf{says}\,\sigma \equiv (A\,\mathbf{says}\,s) \wedge (B\,\mathbf{says}\,\sigma) \qquad \text{(Definition P1)}$$

Principal conjunction is associative, commutative, and idempotent:

$$\vdash (A \wedge B) \wedge C = A \wedge (B \wedge C) \qquad \text{(Axiom P4)}$$
$$\vdash A \wedge B = B \wedge A \qquad \text{(Axiom P4)}$$
$$\vdash A \wedge A = A \qquad \text{(Axiom P4)}$$

Quoting $(B|A)$ is defined as:

$$\vdash (B|A)\,\mathbf{says}\,\sigma \equiv B\,\mathbf{says}\ (A\,\mathbf{says}\,\sigma) \qquad \text{(Definition P2)}$$

In a sense, the quoting operator "curries" a **says** operation from the propositional formula into the principal formula, so that one can talk about a principal quoting another without yet mentioning the specific statement being quoted.

Quoting is associative and distributes over conjunction in both arguments:

$$\vdash (A|B)|C = A|(B|C) \qquad \text{(Axiom P5)}$$
$$\left.\begin{array}{l}\vdash A|(B \wedge C) = (A|B) \wedge (A|C)\\ \vdash (A \wedge B)|C = (A|C) \wedge (B|C)\end{array}\right. \qquad \text{(Axiom P6)}$$

4.2 The "speaks for" relation

A central concept of the calculus is the "speaks for" relation ($\Rightarrow$), which defines a partial order over all principals. This relation encodes the notion of one principal trusting another that we introduced in Section 3.1.2. The statement $B \Rightarrow A$ is read "B speaks for A," and means that whenever B says something, A certainly agrees. Formally, we define

$$\vdash (B \Rightarrow A) \equiv (B = B \wedge A) \qquad \text{(Definition P7)}$$

Why is this the case? If A trusts B, then A says everything B says. So the set of things $B \wedge A$ say must be the same as the set of things B says. It cannot be greater, by its semantic definition in Section 3.1, and it cannot be less, or else there is something B says that A does not.

From the definition we can derive:[5]

$$\vdash (B \Rightarrow A) \supset ((B\,\mathbf{says}\,\sigma) \supset (A\,\mathbf{says}\,\sigma)) \qquad \text{(Theorem P8)}$$

When $B \Rightarrow A$, B is a stronger principal than A in the sense that B can do everything A can do (by making A believe the appropriate performative statement), and perhaps more.

Using the associativity of $\wedge$ for principals, it is clear that $\Rightarrow$ is a transitive relation:

$$\vdash (B \Rightarrow A) \wedge (C \Rightarrow B) \supset C \Rightarrow A \qquad \text{(Theorem L1)}$$

(The $\wedge$ in the theorem is that for statements. We would like to use a different symbol for clarity, but we stick with the notation of Abadi *et al.* here.) Both the $\wedge$ and $|$ operators on principals are monotonic with respect to $\Rightarrow$:

$$\vdash (A \Rightarrow B) \supset ((A \wedge C) \Rightarrow (B \wedge C)) \qquad \text{(Axiom L2)}$$

$$\left.\begin{array}{l}\vdash (A \Rightarrow B) \supset ((A|C) \Rightarrow (B|C))\\ \vdash (A \Rightarrow B) \supset ((C|A) \Rightarrow (C|B))\end{array}\right. \qquad \text{(Axiom L3)}$$

With the speaks-for relation, we can finally see why quoting is a useful operation. One can let $C|B \Rightarrow A$, so that C can only speak for A when it quotes B. Without quoting, we would need a formal accounting for universal quantification over formulas: $\forall \sigma, C\,\mathbf{says}\,B\,\mathbf{says}\,\sigma \supset A\,\mathbf{says}\,\sigma$.

The semantics of $\Rightarrow$ falls out fairly directly. Definition P7 requires that

$$\begin{array}{l}\mathcal{M}, w \models B \Rightarrow A\\ \qquad \textbf{iff } \mathcal{R}(B) = \mathcal{R}(B \wedge A) = \mathcal{R}(B) \cup \mathcal{R}(A)\\ \qquad \textbf{iff } \mathcal{R}(A) \subseteq \mathcal{R}(B)\end{array}$$

[5] Surprisingly, Abadi *et al.* drop Definition P7 and instead treat Theorem P8 as an axiom. Doing so precludes theorems with conclusions containing $\Rightarrow$, since we are left with no axioms with $\Rightarrow$ in the conclusion. In fact, Theorem P8 requires only the weaker operator $\rightarrow$ in its premise, which we discuss in Section **??**.

Notice that the condition on the $\mathcal{R}$ relations is independent of the world w. So the extension function $\mathcal{E}$ is all-or-nothing for speaks-for formulas:

$$\mathcal{E}(B \Rightarrow A) = \begin{cases} W & \text{if } \mathcal{R}(A) \subseteq \mathcal{R}(B) \\ \varnothing & \text{otherwise} \end{cases} \qquad \text{(Definition A1)}$$

4.3 Access Control Lists

The speaks-for relation, because it is transitive, lets us reason broadly about how principals' beliefs affect one another. In the end, however, the server wants to convince itself that some primitive proposition s, perhaps to be interpreted "it is okay to change the contents of the file," is true. To support this, Abadi, Lampson *et al.* use the construct A *controls* s to indicate that principal A's beliefs about s are taken to be truth. It is defined as:

$$A \textit{ controls } s \equiv ((A \,\mathbf{says}\, s) \supset s) \quad \text{(Definition A2)}$$

Now suppose B wants to write to the file that s describes, and the assumptions $\vdash B \Rightarrow A$ and $\vdash A$ *controls* s hold. Then the file server will be able to verify a proof of $\vdash s$, convincing itself that "it is okay to change the contents of the file."

Lampson *et al.* encode access control lists (ACLs) using *controls* assumptions:

$$\text{ACL } (O_1) = \left\{ \begin{array}{l} \vdash A \textit{ controls } s_{read}, \\ \vdash A \textit{ controls } s_{write}, \\ \vdash B \textit{ controls } s_{read} \end{array} \right\}$$

By adjusting which principals' assertions are believed, the ACLs allow or disallow agents to effect action.

4.4 Higher-level operators

The operating system that instantiates the calculus requires resource servers to construct and then verify all necessary proofs [WABL94]. Wobber calls it a *pull* model: it is the servers' job to pull in necessary assumptions and proof components needed to verify an agent's access. Building such proofs, when assumptions include speaks-for formulas with arbitrary combinations of $\wedge$ and $|$ operators, takes exponential time. To make the decision problem tractable, Lampson *et al.* define two high-level operators, **as** and **for**, in terms of the lower-level operators. Each operator is designed to reflect an idiomatic usage pattern of the calculus. The higher-level operators can combine in fewer ways than the lower-level operators, allowing an implementation to exploit characteristics such as associativity and idempotence. In the abstract, the operators can be treated as abbreviations and replaced by their definitions, and they do not affect the calculus. We cover them here to demonstrate the idioms they represent.

4.5 Roles and the "as" operator

Abadi *et al.* define a distinguished, disjoint set of principals called roles. By quoting a role, a principal restricts its own authority. For example, define the roles R_{user} and R_{admin} representing a person acting as a user and as an administrator, respectively. Suppose the ACLs in the system include $A|R_{\text{admin}}$ *controls* s_1 and $A|R_{\text{user}}$ *controls* s_2. In her daily work, Alice may step into her role as user by quoting R_{user}; when she needs to perform administrative tasks, Alice can explicitly quote R_{admin} to gain access to objects such as s_1 that mention her administrative role. More interestingly, Alice can delegate just one of her roles to another principal by arranging that $B \Rightarrow A|R_{\text{user}}$. Now Bob can do anything Alice could do as a user, but he cannot access her administrative resources. Roles can also be used to sandbox untrusted code. When running untrusted software, Alice might delegate to it only authority over $A|R_{\text{untrusted}}$, preventing the code from accessing the bulk of her resources.

The **as** operator stands for quoting when the quoted principal is a role (Axiom R1 in [LABW92]). In a sense, **as** adds strong typing, requiring that its right-hand argument be a role. In contrast to general principals, quoting is idempotent and commutative for roles, and all principals automatically speak for themselves in every role:

$$R|R = R \qquad \forall R \in \mathit{Roles} \quad \text{(Axiom A3)}$$
$$R'|R = R|R' \qquad \forall R, R' \in \mathit{Roles} \quad \text{(Axiom A4)}$$
$$A \Rightarrow A \,\textbf{as}\, R \qquad \forall R \in \mathit{Roles} \quad \text{(Axiom R2)}$$

By virtue of these special features of roles and its strong typing, the **as** operator takes on idempotence and commutativity. This helps make the access control problem tractable.

4.5.1 Semantics for Roles

The axioms above are not supported for general quoting, and yet **as** is simply an abbreviation for quoting. Therefore, the axioms must be justified by some restriction on the possible-worlds relations of the roles themselves. First we define a special principal **1**, the *identity*, who believes everything that is true and nothing that is not:

$$\mathcal{R}(\mathbf{1})(w) = w \qquad \forall w \in W$$

In any given world, **1** considers only that world possible. Therefore, it only tells the truth (the relation is reflexive), and it tells the whole truth (no world has multiple arrows, so it is confused about nothing). The identity serves as the most trusted role a principal can assume. Why? $A \mathbf{as} \mathbf{1}$ is shorthand for $A|\mathbf{1}$, so $\mathcal{R}(A \textbf{ as } \mathbf{1}) = \mathcal{R}(A) \circ \mathcal{R}(\mathbf{1}) = \mathcal{R}(A)$: the identity role does not limit A's authority at all.

All roles are principals whose relations are constrained as follows:

$$\mathcal{R}(R_1) \subseteq \mathcal{R}(\mathbf{1})$$

This means that the role relation may contain some edges $\langle w, w \rangle$ and not others, but no edges that take one world to another world. A role, when composed with another principal's relation, cannot expand the set of worlds the principal considers possible, only reduce it. See Figure 2 for an illustration.

We are now prepared to justify the axioms for roles. The first property is idempotence. $\mathcal{R}(R_1)$ takes each world to either itself or nowhere, so composing $\mathcal{R}(R_1)$ with itself should do the same. The second property is commutativity. An arrow appears in $\mathcal{R}(R_1) \circ \mathcal{R}(R_2)$ exactly when it appears in $\mathcal{R}(R_1) \cap \mathcal{R}(R_2)$, and $\cap$ is commutative. Finally, $A \Rightarrow A \textbf{ as } R_1$ is automatically true when R_1 is a role. Why? Composing $\mathcal{R}(R_1)$ onto $\mathcal{R}(A)$ cannot introduce any new worlds (since the arrows of $\mathcal{R}(R_1)$ are all reflexive), but may eliminate worlds (when $\mathcal{R}(R_1)(w) = \varnothing$). Hence

$$\mathcal{R}(A) \circ \mathcal{R}(R_1) \subseteq \mathcal{R}(A)$$

and we conclude $A \Rightarrow A \textbf{ as } R_1$.

4.6 Delegation and the "for" operator

Besides encoding roles, quoting can be used to encode delegations to trusted principals in a restricted way. Here is the problem: Imagine that both Alice and Bob log in to machine M. Using just the speaks-for operator, Alice might establish that $M \Rightarrow A$ and Bob that $M \Rightarrow B$. But then when Bob (sitting at his terminal to machine M) tries to read a file that only A has permission to read, M would say the request, and the server would reason that A believed it. In this situation, the access-control system cannot help the server reason about whether the file should be read, since M has not provided enough information.

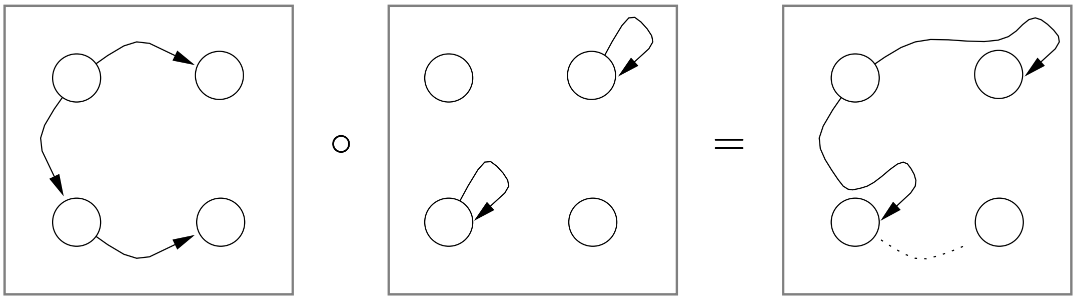
Instead, A could require that M explicitly mention A whenever it makes requests on A's behalf: $M|A \Rightarrow A$. Now when M is working for B, it will be quoting B, not A, and A's file is safe. If M were corrupt, of course, it could still abuse the authority granted it by A. But quoting principals helps an honest M pass the right information to resource servers for access-control decisions.

Lampson *et al.* define a slightly more complicated concept of delegation from A to B, written as the compound principal B **for** A. The key idea behind delegation is that both the delegator A and the delegate B must take some explicit action for the delegation to take effect:

$$A \textbf{ says } B|A \Rightarrow (B \textbf{ for } A)$$
$$B|A \textbf{ says } B|A \Rightarrow (B \textbf{ for } A)$$

from which, using the definition of **for** in [LABW92, p. 295], we can conclude

$$(B|A) \Rightarrow (B \textbf{ for } A)$$



An arbitrary principal relation $\mathcal{R}(A)$ ...

... composed with a role relation $\mathcal{R}(R)$ ...

... gives a new relation that is always a subset of $\mathcal{R}(A)$.

Figure 2: *Roles reduce relations they are composed with.*

Then A installs B **for** A in ACLs for any resources it wishes to allow B to access on its behalf.

The difference between B simply taking care to always quote A and B receiving a delegation to B **for** A is subtle. In both cases, A must explicitly hand off authority to B. And in both cases, B has to take some explicit action to accept the delegation; in the first case, that action is to quote A, in the second, it must also make a separate statement accepting the delegation.

Like **as**, **for** seems to be introduced for its special properties, to enable a more efficient pull-style theorem-proving implementation.

We have completed our review presentation of the calculus due to Abadi, Lampson *et al.*

5 Review: The Simple Public Key Infrastructure

The Simple Public Key Infrastructure 2.0 (SPKI, pronounced "spooky") is an Internet Experimental Protocol created by Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen [EFL+99]. As its name suggests, it is designed to be a unifying standard for supporting public key authorization across the global Internet. We highlight here some of the features of SPKI relevant to this work.

First, SPKI's primary goal is to provide a server with evidence that the holder of a given cryptographic key is ultimately authorized for a request signed by that key. This goal contrasts with that of other public-key infrastructure efforts that attempt to bind keys to identities, and leave authorization to be handled in the conventional fashion by ACLs that map identity to authorization.

In this section, we review the types of certificates that SPKI supports, and outline the procedure used to determine whether a given certificate chain supports a requested operation.

5.1 Certificate types

SPKI defines its own certificate format, as well as an internal representation of certificates to which it can map other inputs, such as PGP certificates, X.509 certificates, or locally maintained ACL entries. Authorization results can be constructed from inputs providing information in one of three forms:

- $\langle$authorization, key$\rangle$
- $\langle$authorization, name$\rangle$
- $\langle$name, key$\rangle$

The first form coincides with SPKI's design philosophy of mapping keys directly to authorizations. Inputs of the latter two forms must ultimately be combined to form a $\langle$authorization, key$\rangle$ mapping to become useful.

Inputs of the first two forms are mapped into a data structure called a *5-tuple* for internal processing; inputs of the latter form are mapped into a data structure called a *4-tuple*.

5.2 The SPKI 5-tuple

A 5-tuple has the following fields:

- issuer: the public key granting the permission defined by the 5-tuple
- subject: a public key or name to which the permission is being granted
- delegation-control: a boolean value indicating whether this permission may be further delegated
- authorization: a set of primitive permissions being granted
- validity dates: a date range limiting the validity of this delegation

The intended meaning is that the issuer grants the subject the permission described in the authorization field for the duration of the validity dates. If the delegation-control bit is set, the subject may further delegate any or all of the permission to another subject.

The subject in a 5-tuple (or a 4-tuple, which we present shortly) may be a k-of-n threshold function. In this case, the permission is delegated to any principal that can prove it is authorized to speak for any k of the n "subordinate" subjects listed in the threshold function.

The authorization fields contain primitive permissions whose interpretation is left to the application employing the SPKI authorization engine. These permissions are represented using *auth tags*. Tags encode infinitely large sets of primitive statements in a form that permits a compact representation of certain subsets. Notably, a tag can represent only a set of primitive symbols; never a formula made from the negation or conjunction of primitive symbols. Tags admit a simple intersection algorithm that always yields a compact representation of the intersected set.

SPKI certificates may also indicate an on-line mechanism for verifying that the issuer considers a certificate still valid. Two of the checks, the certificate revocation list (CRL, a negative list of revoked certificates) and the timed revalidation (a positive list of still-valid certificates), are performed by consulting a list revised more frequently than the original certificate being checked. The one-time revalidation check, which "represents a validity interval of zero" [EFL+99, p. 21], is performed by contacting the specified server to verify that the server still approves the certificate.

5.3 The SPKI 4-tuple

Symbolic names are always interpreted relative to a globally unambiguous name, usually a public key. As a consequence, the definition of a symbolic name is never ambiguous; it is always the definition supplied by the key that grounds the name. The SPKI authors contrast this situation with that of PGP, where symbolic names reside in a global namespace, and their meaning depends on the beholder and the "introducers" that the beholder trusts.

A symbolic name ultimately is defined as one or more keys, although a single 4-tuple may define a name in terms of a chain of other names grounded in a key. In that case, other 4-tuples must participate in the reduction of the name chain to a final key. A 4-tuple has the following fields:

- issuer: the public key defining this name in its private name space.
- name: the name being defined
- subject: a public key or name to which the name is bound.
- validity dates: a date range limiting the validity of this delegation

The intended meaning of a 4-tuple is that the issuer defines the symbolic name, when grounded by the issuer's key, to be equal to the key identified by the subject for the duration of the validity dates. It is easy to read this definition backwards. Note that a name definition tuple does not give the issuer control over the subject, but the subject control over any permission elsewhere granted to the grounded name "issuer: name." Hence a threshold subject is also meaningful as the subject of a 4-tuple; its use means that if a principal speaks for k of the n subordinate subjects, that principal also speaks for "issuer: name," and hence garners any permission granted to that name.

5.4 Tuple reduction

The SPKI access-control decision procedure is called "tuple reduction." Once the appropriate certificates for an access-control decision have been gathered, the on-line checks performed, and the certificates converted into internal tuples, the tuples are "reduced." If the reduction results in a 5-tuple issued by the server that grants the requested permission to the key that signed the request, then the request is authorized.

Reduction proceeds as follows. First, 4-tuples are reduced to resolve names. 4-tuples that define a name in terms of another grounded chain of names are reduced using 4-tuples that define a name in terms of a key. Eventually, 4-tuples of the former form are reduced to 4-tuples of the latter form. The validity date stored in the outcome of each reduction is the intersection of the validity dates of the 4-tuples being reduced.

Then the $\langle \text{name}, \text{key} \rangle$ bindings formed by the reduced 4-tuples are applied to resolve names in 5-tuples back to keys, again carrying validity dates through with intersection operations. This operation turns $\langle \text{authorization}, \text{name} \rangle$ 5-tuples into $\langle \text{authorization}, \text{key} \rangle$ tuples.

At this point, each 5-tuple represents a subject key (or threshold subject defined as a set of keys) with authorization to perform some set of actions on behalf of the issuer key. When two 5-tuples form a chain of delegation (the issuer of the second is the subject of the first, and the first tuple allows further delegation), the 5-tuples are reduced to a new tuple whose subject is the subject of the second tuple and whose issuer is the issuer of the first. The reduced tuple carries the intersection of the authorizations of the source tuples as its authorization, and the intersection of the validity dates of the source tuples as its validity dates. Finally, the reduced tuple carries the same delegation control bit as the second tuple did. Think of the delegation control bit as the coupling on the back of a boxcar; if the first tuple lacks it, the cars cannot couple; if the second tuple lacks it, the cars may couple, but the resulting "super-car" will also lack a rear coupling.

Set	*Example members*	*Description*
Σ	s, t	The set of primitive propositions. They represent resources.
Σ^*	σ, τ $s \wedge t$	The set of well-formed formulas (statements) constructed from Σ, $\wedge$, $\neg$, $\mathcal{A}\,\mathbf{says}$, and $\mathcal{B} \Rightarrow \mathcal{A}$
2^{Σ^*}	S, T, V	The set of sets of statements
P	A, B	The set of primitive principals. They represent agents, including people, machines, programs, and communications channels.
P^*	$\mathcal{A}, \mathcal{B}$ $A \wedge B$	The set of compound principals constructed from P, $\wedge$, $\mid$, and $\cdot N$
$\mathcal{N}$	N	The set of local names

Figure 3: *The symbols used to represent sets in this paper.*

We return to SPKI in Section 9, where we apply our extended calculus to model SPKI.

6 The logic and semantics of restricted delegation

Lampson *et al.* mention in passing the idea of a qualified speaks-for operator [LABW92, p. 272]. In this section, we introduce our *speaks-for-regarding* operator, which formalizes the notion of the restricted speaks-for operator. It is written $B \overset{T}{\Rightarrow} A$, and read "$B$ speaks for A regarding the set of statements in T." T is any subset of Σ^*. The desired meaning is that when $\sigma \in T$,

$$B \overset{T}{\Rightarrow} A \supset ((B\,\mathbf{says}\,\sigma) \supset (A\,\mathbf{says}\,\sigma))$$

The power of the speaks-for-regarding operator $\overset{T}{\Rightarrow}$ is that A can delegate a subset of its authority *without modifying any ACLs.* Contrast the situation with the use of roles in Section 4.5, where to delegate authority over a restricted subset of her resources, a user had to define a role and install that role in the ACLs of each resource to be shared.

Restricted speaks-for is transitive:

$$\vdash (\mathcal{C} \overset{T}{\Rightarrow} \mathcal{B}) \land (\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}) \supset (\mathcal{C} \overset{T}{\Rightarrow} \mathcal{A}) \quad \text{(Axiom E1)}$$

We expect the $\land$ operation on principals to be monotonic over $\overset{T}{\Rightarrow}$:

$$\vdash (\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}) \supset (\mathcal{B} \land \mathcal{C}) \overset{T}{\Rightarrow} (\mathcal{A} \land \mathcal{C}) \quad \text{(Axiom E2)}$$

Restricted control over two principals is the same as restricted control over their conjunct:

$$\vdash (\mathcal{C} \overset{T}{\Rightarrow} \mathcal{A}) \land (\mathcal{C} \overset{T}{\Rightarrow} \mathcal{B}) \equiv \mathcal{C} \overset{T}{\Rightarrow} (\mathcal{A} \land \mathcal{B}) \quad \text{(Axiom E3)}$$

Let $\mathcal{U}$ be the universe of all well-formed formulas; that is, those formulas over which a model $\mathcal{M}$ defines $\mathcal{E}$. Restricted speaks-for degenerates to the original speaks-for when the restriction set is the set of all statements:

$$\vdash (\mathcal{B} \overset{\mathcal{U}}{\Rightarrow} \mathcal{A}) \equiv (\mathcal{B} \Rightarrow \mathcal{A}) \quad \text{(Axiom E4)}$$

If Bob speaks for Alice regarding a set of statements T, he surely speaks for her regarding a subset $T' \subseteq T$:

$$\forall T' \subseteq T, \quad \vdash (\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}) \supset (\mathcal{B} \overset{T'}{\Rightarrow} \mathcal{A}) \quad \text{(Axiom E5)}$$

Using Axiom E5, a chain of delegations can be collapsed to a single delegation, connecting the head principal in the chain to the tail, whose restriction set is the intersection of the restriction sets of each of the original delegations.

$$\vdash (\mathcal{C} \overset{S}{\Rightarrow} \mathcal{B}) \land (\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}) \supset (\mathcal{C} \overset{S \cap T}{\Rightarrow} \mathcal{A}) \quad \text{(Theorem E6)}$$

This is not to say that $\mathcal{C}$ may not speak for $\mathcal{A}$ regarding more statements than those in the intersection; we address this topic further in Section 9.9.

If we have two restricted delegations from Alice to Bob, we might expect Alice to speak for Bob with respect to the union of the restriction sets. Because of the semantics we choose for $\overset{T}{\Rightarrow}$, however, this intuition does not hold.

$$(B \overset{S}{\Rightarrow} A) \land (B \overset{T}{\Rightarrow} A) \not\supset B \overset{S \cup T}{\Rightarrow} A \quad \text{(Result E7)}$$

In Section **??**, we describe a relation weaker than $\overset{T}{\Rightarrow}$ for which the intuitive statement holds.

The quoting operator on principals ($|$) is monotonic in both arguments over $\Rightarrow$. Quoting is still monotonic over $\overset{T}{\Rightarrow}$ in its left argument:

$$\vdash \left(\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}\right) \supset \mathcal{C}|\mathcal{B} \overset{T}{\Rightarrow} \mathcal{C}|\mathcal{A} \quad \text{(Axiom E8)}$$

Our semantics does not justify monotonicity in the right argument, however:

$$\left(\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}\right) \not\supset \mathcal{B}|\mathcal{C} \overset{T}{\Rightarrow} \mathcal{A}|\mathcal{C} \quad \text{(Result E9)}$$

This result appears to limit the usefulness of quoting. The same counterexample that shows Result E9 shows the same property for the weak speaks-for relation defined in Section **??**, so it seems that the notion of quoting simply does not mix easily with restricted delegation.

We can, however, propagate the quoted principal through the restriction set. Let T^* be the closure of T with respect to the propositional operators $\neg$ and $\land$: $T \subseteq T^*$, and if $\sigma, \tau \in T^*$, then $\neg\sigma \in T^*$ and $\sigma \land \tau \in T^*$. Furthermore let TC be the closure of T with respect to the modal operator $\mathcal{C}\,\mathbf{says}$: $T \subseteq TC$, and if $\sigma \in TC$, then $(\mathcal{C}\,\mathbf{says}\,\sigma) \in TC$. Now $(T^*)C$ is the modal closure applied to the propositional closure of some original set T. With these definitions, we can justify this axiom:

$$\vdash \left(\mathcal{B} \overset{(T^*)C}{\Rightarrow} \mathcal{A}\right) \supset \left(\mathcal{B}|\mathcal{C} \overset{T}{\Rightarrow} \mathcal{A}|\mathcal{C}\right) \quad \text{(Axiom E10)}$$

When $T = \mathcal{U}$, this axiom reduces to showing right-monotonicity for the original speaks-for relation. This axiom means that $\mathcal{A}$'s restricted delegation to $\mathcal{B}$ must explicitly include any "quotes" of $\mathcal{C}$ that it is willing to believe $\mathcal{B}$ about. It seems awkward, but it is a useful result. Why? Because in any possible-worlds semantics wherein $(\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A}) \supset (\mathcal{B}|\mathcal{C} \stackrel{T}{\Rightarrow} \mathcal{A}|\mathcal{C})$ for *all* principals $\mathcal{C}$, the relation representing $\mathcal{A}$ depends on every other principal relation. The introduction of malicious principals with cleverly-chosen relations into such a system can effectively expand T until $T = \mathcal{U}$.

6.1 Semantics of $\stackrel{T}{\Rightarrow}$

We use a semantics based on possible worlds, modeling a system with a *model* $\mathcal{M} = \langle W, w_0, I, J\rangle$ whose components are defined as in [ABLP93]. The semantic definition of $\stackrel{T}{\Rightarrow}$ is based on the notion of "projecting" a model into a space where only the statements in set T are relevant. The idea behind this definition is that if one were to take the "quotient" of a model M with respect to the dual of T, the resulting model $\overline{M}$ would be concerned only with statements in T. $B \Rightarrow A$ in $\overline{M}$ should be equivalent to $B \stackrel{T}{\Rightarrow} A$ in the original model. The model $\overline{M}$ is a projection of M that only preserves information about statements in T.

We begin the construction by defining an equivalence relation $\cong_T$: $W \times W$ that relates two worlds whenever they agree on all statements in T:

$$w \cong_T w' \ \textbf{iff} \ \big(\forall \sigma \in T, w \in \mathcal{E}(\sigma) \ \textbf{iff} \ w' \in \mathcal{E}(\sigma)\big) \qquad \text{(Definition E11)}$$

Then we define the mapping $\phi_T : W \rightarrow \overline{W}$ that takes worlds from the original model to equivalence classes under $\cong_T$:

$$\phi_T(w) = \phi_T(w') \ \textbf{iff} \ w \cong_T w' \qquad \text{(Definition E12)}$$

The equivalence classes belong to a set $\overline{W} = 2^T$; notice that worlds (equivalence class representatives) in $\overline{M}$ cannot be confused with those in M. We give a construction of $\phi_T(w)$ in Appendix Section A.1.

Next we extend ϕ_T to the function $\phi_T^w : 2^W \rightarrow 2^{\overline{W}}$ that maps a set of worlds $S_w \subseteq W$ to a set of equivalence class representatives in the projected model:

$$\phi_T^w(S_w) = \{\overline{w} \mid \exists w \in S_w,\ \overline{w} = \phi_T(w)\} \qquad \text{(Definition E13)}$$

We use bar notation ($\overline{w}$) to indicate an equivalence class representative (member of a world of a projected model) as opposed to a member of W in the original model.

We can now give our semantic definition of restricted delegation:

$$\mathcal{E}(\mathcal{B} \stackrel{T}{\Rightarrow} A) = \begin{cases} W & \text{if } \forall w_0 \begin{pmatrix} \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) \subseteq \\ \phi_T^w(\mathcal{R}(\mathcal{B})(w_0)) \end{pmatrix} \\ \varnothing & \text{otherwise} \end{cases} \qquad \text{(Definition E14)}$$

For the justifications of several of the axioms it is more convenient to shift the projection (ϕ) operation to one side of the subset relation. To do so, we define

$$\phi_T^+(R) = \big\{\langle w_0, w_1'\rangle \mid \exists w_1 \cong_T w_1',\ \langle w_0, w_1\rangle \in R\big\} \qquad \text{(Definition E15)}$$

Think of ϕ_T^+ as a function that introduces as many edges as it can to a relation without disturbing its projection under T.

We can use ϕ_T^+ to give an equivalent definition of $\stackrel{T}{\Rightarrow}$:

$$\mathcal{E}(\mathcal{B} \stackrel{T}{\Rightarrow} A) = \begin{cases} W & \text{if } \mathcal{R}(\mathcal{A}) \subseteq \phi_T^+(\mathcal{R}(\mathcal{B})) \\ \varnothing & \text{otherwise} \end{cases} \qquad \text{(Definition E16)}$$

The symbolic gymnastics of moving the projection to the right side of the $\subseteq$ relation is equivalent to the definition in terms of ϕ_T^w, but it makes some of the proofs more concise. The equivalence is shown in Appendix A.2.

A casual intuition for this definition is that ϕ_T projects from the full model M down to a model in which worlds are only distinguished if they differ with regard to the truth of statements in T. If we collapse away the accessibility arrows that do not say anything about what is happening in T, and A's relation is a subset of B's relation in the projection, then A knows everything B knows about statements in T. This intuition is exactly what we want for restricted delegation.

What happens if we take an alternative semantic definition for restricted delegation?

6.2 Additional benefits of $\stackrel{T}{\Rightarrow}$

Introducing the $\stackrel{T}{\Rightarrow}$ operator to the logic not only provides the important feature of restricted delegation, but it simplifies the logic by replacing the *controls* operator, replacing roles, and providing a formal mechanism for the treatment of expiration times.

6.2.1 Supplanting *controls*

Now that we have the restricted speaks-for relation, we can dispense with the special *controls* operator for building ACLs.

Recall Abadi *et al.*'s special identity principal **1** from Section 4.5.1. Because it believes only truth, $(\mathbf{1}\,\mathbf{says}\,s) \supset s$ for all statements s. That is, there is an implicit principal that controls all statements. We can replace every statement of the form A *controls* s with an equivalent one: $A \stackrel{\{s\}}{\Rightarrow}_{w_0} \mathbf{1}$. This statement ensures that if $A\,\mathbf{says}\,s$, then at the actual world w_0 of the model, $\mathbf{1}\,\mathbf{says}\,s$. Since the **1** relation only contains edges from a node to itself, this condition can only be satisfied by selecting an actual world w_0 where s is true.

6.2.2 Supplanting roles

Roles as originally defined are attractive, but they have the significant difficulty that introducing a new restricted role R_2 involves finding all of the objects that role should be allowed to touch, and adding $A\,\mathbf{as}\,R_2$ to each of those ACLs. When one of those objects does not allow ACL modifications by A, it is impossible for A to express the desired new role. The SPKI document gives a vivid example that shows how ACL management can become unwieldy [EFL+99, p. 17].

With the speaks-for-regarding relation, A can introduce a new role R_2 for itself by allowing $(A\,\mathbf{as}\,R_2) \stackrel{T_2}{\Rightarrow} A$. In fact, roles are no longer necessary at all, but the **as** and **for** operator, or operators like them, may still be useful for building tractable implementations.

Roles, as semantically defined by Abadi *et al.*, can also have surprising consequences because they belong to a global "namespace." Imagine that both Alice and Bob use the role R_{user} in their ACLs. That means that the same relation $\mathcal{R}(R_{\text{user}})$ encodes both the way that $A\,\mathbf{as}\,R_{\text{user}}$ is weaker than A, and the way that $B\,\mathbf{as}\,R_{\text{user}}$ is weaker than B.

6.2.3 Formalizing statement expiration

Lampson *et al.* treat expiration times casually in [LABW92, p. 270]: "Each premise has a *lifetime*, and the lifetime of the conclusion, and therefore of the credentials, is the lifetime of the shortest-lived premise." It is likely that a formal treatment of lifetimes would be time-consuming and unsurprising, but the lifetimes are an unsightly element glued onto an otherwise elegant logical framework. Fortunately, the $\stackrel{T}{\Rightarrow}$ relation allows us to dispense with lifetimes.

Recall from Section 4.3 that the primitive statements such as s are meant to encode some operation in a real system. Assume that each s describes not only an operation, but the effective time the operation is to take place.[6] Further, assume a restriction set T in a delegation $B \stackrel{T}{\Rightarrow} A$ includes restrictions on the times of the operations under consideration. After the last time allowed by the set, the delegation remains logically valid, but becomes useless in practice. Furthermore, restrictions on T can be more than expiration times; one can encode arbitrary temporal restrictions, such as only allowing a delegation to be valid on Friday afternoons.

[6] Like Lampson *et al.*, we ignore the issue of securely providing loosely synchronized clocks.

7 The semantics of SPKI names

Recall from Section 6.2.2 how roles share a global "namespace," and the danger of crosstalk between applications of the same role. SPKI names are promising, but they have the same property: identical names have different meaning depending on the "scope" in which they appear. To model names, we need to extend our logic and semantics.

We introduce to the logic a new set of primitive *names*, $\mathcal{N}$. We also extend principal expressions to include those of the form $P \cdot N$, where P is an arbitrary principal expression and $N \in \mathcal{N}$. $P \cdot N$ is read "P's N." Because $\cdot$ only accepts a principal as its left argument, there is no ambiguity in the order of operations; $P \cdot N_1 \cdot N_2$ can only be parenthesized $(P \cdot N_1) \cdot N_2$.

7.1 The logic of names

What properties do we want names to have?

Local namespaces. First, a principal should control the meaning of any names defined relative to itself:

$$\forall \text{ principals } \mathcal{A}, \text{ names } N:$$
$$(\mathcal{A} \,\mathbf{says}\, (\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A} \cdot N)) \supset (\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A} \cdot N)$$

We do not take this statement as an axiom for the same reason that Abadi, Lampson *et al.* do not accept the handoff axiom [LABW92, p. 715], [ABLP93, p. 273]. In particular, our semantics does not support it. Instead, as with the handoff axiom, we allow the implementation to assume appropriate instances of it.

Left-monotonicity. Second, name application should be monotonic over speaks-for. If Alice binds her name "barber" to Bob, and Bob binds his name "butcher" to Charlie, then we want "Alice's barber's butcher" to be bound to Charlie.

$$(\mathcal{B} \Rightarrow \mathcal{A}) \supset (\mathcal{B} \cdot N \Rightarrow \mathcal{A} \cdot N) \qquad \text{(Axiom E17)}$$

Using this rule, we can write the following to capture the desired intuition:

$$(B \Rightarrow A \cdot N_{\text{barber}}) \supset$$
$$B \cdot N_{\text{butcher}} \Rightarrow A \cdot N_{\text{barber}} \cdot N_{\text{butcher}}$$

Distributivity. We combine the following pair of results

$$(\mathcal{A} \wedge \mathcal{B}) \cdot N \Rightarrow (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \qquad \text{(Theorem E18)}$$
$$(\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \Rightarrow (\mathcal{A} \wedge \mathcal{B}) \cdot N \qquad \text{(Axiom E19)}$$

to show that names distribute over principal conjunction:

$$(\mathcal{A} \wedge \mathcal{B}) \cdot N = (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \qquad \text{(Theorem E20)}$$

Here is a motivating example: If Alice has two doctors Ed (E) and Fred (F), and Bob visits doctors Fred and George (G), then who is "(Alice and Bob)'s doctor?" Fred is the only person who serves as both people's doctor.

No quoting axiom. The principal $(A|B) \cdot N$ can be written, but we have yet to find a meaningful intuitive interpretation for it. $(A|B) \cdot N$ bears no obvious relation to $(A \cdot N)|(B \cdot N)$, for example. We allow the principal in our logic, but we have no axioms for extracting quoting from inside a name application.

Nonidempotence. Finally, application of names should not be always idempotent. Unless some other speaks-for statement causes it, there is no reason that "Bob's barber's barber" should speak for "Bob's barber." We were initially tempted to model name application ($\cdot$) with role application, because roles satisfy Axiom E17; however, roles are idempotent.

7.2 The semantics of names

We mentioned above that names and name application cannot be modeled with the roles and the quoting operator, because quoting a role is always idempotent. Furthermore, using the same role for multiple uses of the same name by different principals introduces crosstalk as described in Section 6.2.2.

Instead, we model names as follows. First, we add a new element K to the tuple that defines a model. A model with naming consists of:

$$\mathcal{M} = \langle W, w_0, I, J, K \rangle$$

The new interpretation function $K : P \times \mathcal{N} \rightarrow 2^{W \times W}$ maps a primitive principal A and a name N to a relation. The idea is that principals only define the first level of names in their namespaces; all other names are consequences of chained first-level name definitions.

Next we extend $\mathcal{R}$ to define the relations for principals formed through name application. We want to define $\mathcal{R}(\mathcal{A} \cdot N)$ as the intersection of several other sets, each requirement ensuring a desired property. Our definition, however, would end up circular (at requirement I, with equal principals) if it were expressed in terms of set intersection. Instead, we define $\mathcal{R}(\mathcal{A} \cdot N)$ as the largest relation (subset of $2^{W \times W}$) satisfying all of the following requirements:

$$\begin{aligned}
\mathcal{R}(\mathcal{A} \cdot N) \subseteq & \mathcal{R}(\mathcal{B} \cdot N) && \text{(I)} \\
& && (\forall \mathcal{B} : \mathcal{R}(\mathcal{A}) \subseteq \mathcal{R}(\mathcal{B})) \\
\mathcal{R}(\mathcal{A} \cdot N) \subseteq & K(\mathcal{A}, N) && \text{(II)} \\
& && (\text{when } \mathcal{A} \in P) \\
\mathcal{R}(\mathcal{A} \cdot N) \subseteq & \mathcal{R}(\mathcal{B} \cdot N) \cup \mathcal{R}(\mathcal{C} \cdot N) && \text{(III)} \\
& && (\text{when } \mathcal{A} = \mathcal{B} \wedge \mathcal{C}) \\
& && (\text{Definition E21})
\end{aligned}$$

Requirement I supports Axiom E17. Requirement II applies only to primitive principals, and allows each primitive principal to introduce definitions for first-level names in that principal's namespace. A system implementing instances of the handoff rule does so conceptually by modifying $K(\mathcal{A}, N)$. Requirement III only applies to principal expressions that are conjunctions, and justifies Theorem E20.

There is no question some such largest relation exists. Since each requirement is a subset relation, at least the empty set satisfies all three. There is an upper bound, since every relation is a subset of the finite set $W \times W$.

7.3 Abadi's semantics for linked local namespaces

Abadi gives an alternate logic and semantics for SPKI-style linked local namespaces [Aba98]. (He refers to SDSI, from which SPKI 2.0 derives.) Abadi's notation diverges from that used in [ABLP93], but the semantics are the same. Figure 4 helps translate the notation. Our semantics differs in three interesting ways.

First, SPKI has special global names, so that if N_G is a global name, $\mathcal{A} \cdot N_G = N_G$. The result is that the same syntactic construct can be used to bind a local name to another local name or to a globally-specified name. All names in linking statements are implicitly prefixed by the name of the speaking principal; but if the explicitly mentioned name is global, the prefix has no consequence. We consider this syntactic sugar, and leave it to an implementation to determine from explicit cues (such as a key specification or a SDSI name that ends in !!) whether a mentioned principal should be interpreted as local to the speaker.

Second, Abadi's logic adopts the handoff rule for names, which he calls the "Linking" axiom Here it is, translated to our terminology:

$$\mathcal{A}\,\textbf{says}\,(\mathcal{B} \Rightarrow (\mathcal{A} \cdot N)) \supset (\mathcal{B} \Rightarrow (\mathcal{A} \cdot N))$$

He validates the axiom by the use of composition to model name application, with which we disagree.

The third and most important way our semantics differs from Abadi's is that Abadi's semantics models name application as quoting (composition). Each unqualified (local) name is mapped to a single relation. This property can introduce crosstalk between otherwise unconnected principals; recall the example from Section 6.2.2. Even when a name relation is not constrained to be a role, the same problem arises. For example, let N represent the name "doctor." Imagine that Bob assigns Charlie to be his doctor: $C \Rightarrow B|N$. This is fine; Charlie should be able to do some things on Bob's behalf, but not everything: If $B|N \overset{T}{\Rightarrow} B$, then Charlie can do the things in T.

Enter Alice, who is not only omniscient ($A = \mathbf{1}$), but serves as her own doctor ($A \Rightarrow A|N$).

Abadi's notation	Our notation
S	Σ
$\mu : S \times \mathcal{W} \to \{\mathit{true}, \mathit{false}\}$	$I : \Sigma \to 2^W$
$\rho : N \times \mathcal{W} \to 2^{\mathcal{W}}$	$K : P \times \mathcal{N} \to 2^{W \times W}$
$a \in \mathcal{W}$	$w \in W$
principals p, q	$\mathcal{A}, \mathcal{B} \in P^*$
$n \in N$	$N \in \mathcal{N}$
$[\![n]\!]_a = \rho(n, a)$	$\mathcal{R}(\mathcal{A} \cdot N)(w) = K(\mathcal{A}, N)(w)$
$[\![p'\mathtt{s}\ n]\!]_a$	$\mathcal{R}(\mathcal{A} \cdot N)(w)$

Figure 4: *A guide to translating between Abadi's notation and ours*

Abadi's semantics requires that $\mathcal{R}(\mathbf{1}) \circ \mathcal{R}(N) \subseteq \mathcal{R}(\mathbf{1})$. At worst, $\mathcal{R}(N) = \mathcal{R}(\mathbf{1})$, causing $B|N = B$, enabling Charlie's doctor to make investment decisions on Charlie's behalf. At best, $\mathcal{R}(N) \subset \mathcal{R}(\mathbf{1})$, and $B|N$ begins spouting off random statements, some of which may be in T, making Bob believe random statements.

Our semantics escapes this fate by assigning to each use of a name its own relation, then ensuring the correct subset relationships remain among those relations. We must admit that our semantics for names is at best opaque. Although using an existential definition like "largest set satisfying the requirements" is not illuminating, we feel it is better than the alternative.

8 The semantics of authorization tag notation

An important part of SPKI is a user-defined `<tag>` object that describes either a specific request to be verified, or a set of permissions granted in a delegation. SPKI defines how tags are to be intersected, which gives the user some idea about the meaning of tags. In this chapter, we derive a formal language for tags and show that tags are (almost) closed under intersection. We also show that tags have a desirable property that we depend upon for our formalization of SPKI's properties in Section 9.9.

8.1 Overview

One confusing aspect of tags is that they serve to represent both single requests and delegation restrictions (sets of requests to be permitted). How do they do this? The short answer is containment. Most tags represent an infinite set of finite strings. Let us call those strings *powers*. A request to be verified is a set of powers; the intuition is that executing the request requires that the requester have at least a certain set of privileges. A permission describes another set of powers. If the permission's set contains the request's set, then the permission grants at least each of the powers required for the request.

Why use infinite sets of powers? For extensibility. In general, a permission (tag) is represented by an infinite set of powers. Therefore, it can always be further subdivided into more-specific permissions, each still represented by an infinite subset of the original permission's powers.

Given this motivation for the structure of tags, we construct tags from the ground up using grammars. We begin by defining bytestrings and the atomic finite strings we have called powers.

8.2 Bytestrings and powers

Let Σ be the natural alphabet of our system; in our case, let it be the set of 256 octets. Let $\mathbf{B}$ be the set of all finite-length strings of octets, Σ^*:

$$\begin{aligned} \mathbf{B} :=\, & \sigma\mathbf{B} \qquad\qquad (\forall\sigma \in \Sigma) \\ & |\varepsilon \end{aligned}$$

This is the set SPKI calls "bytestrings." Next we extend Σ with three metasymbols to unambiguously demarcate the list structure of a power: $\Sigma' = \Sigma \cup \{\llparenthesis\,, \sqcup\,, \rrparenthesis\}$. Now we define the mutually-recursive sets of expressions (**E**), lists of expressions (**L**), and non-empty lists of expressions (**N**):

$$
\begin{aligned}
\mathbf{E} :=& \mathbf{B} \\
&| \llparenthesis\, \mathbf{L} \,\rrparenthesis \\
\mathbf{L} :=& \mathbf{N} \\
&| \varepsilon \\
\mathbf{N} :=& \mathbf{E} \sqcup \mathbf{N} \\
&| \mathbf{E}
\end{aligned}
$$

The gymnastics with the non-empty lists serve to prevent lists from ending with a $\sqcup$ just before the $\rrparenthesis$; this feels right but it is not important for the development of the semantics.

We call the set of expressions **E** the set of powers. Notice that every power is a finite object with an unambiguous tree structure defined by the special delimiters $\{\llparenthesis\,, \sqcup\,, \rrparenthesis\}$. Every internal node is a list, and every leaf node is a bytestring.

8.3 Auths

Recall that a tag, which specifies either a request or a set of delegated permissions, represents a (usually infinite) set of powers. Our next task is to define the set of *auths*, each a subset of **E**. In the next section, we show that the set of SPKI tags is, with some caveats, isomorphic to the set of auths we define here.

We first define the base-case auths:

$$A_{null} = \varnothing \qquad \text{(Definition T1)}$$

$$A_* = \mathbf{E} \qquad \text{(Definition T2)}$$

$$A_{bs}(b) = \{b\} \qquad (\forall b \in \mathbf{B}) \qquad \text{(Definition T3)}$$

$$A_0 = A_{null} \bigcup A_* \bigcup_{b \in \mathbf{B}} A_{bs}(b) \qquad \text{(Definition T4)}$$

Then we extend it recursively:

$$
\begin{aligned}
&A_{set}(a_1 \ldots a_k) \\
&\quad = \bigcup_{1 \le i \le k} a_i \qquad \text{(Definition T5)}
\end{aligned}
$$

$$
\begin{aligned}
&A_{list}(a_1 \ldots a_k) \\
&\quad = \left\{ \begin{array}{l}
\llparenthesis\, x_1 \sqcup x_2 \sqcup \ldots \sqcup x_k \sqcup \cdots \sqcup x_n \,\rrparenthesis \\
\text{such that} \\
\qquad x_i \in a_i \quad \forall i, 1 \le i \le k \\
\qquad x_i \in \mathbf{E} \quad \forall i, k < i
\end{array} \right\} \\
&\qquad \text{(Definition T6)}
\end{aligned}
$$

$$
\begin{aligned}
A_j = \quad & A_{j-1} \\
\bigcup \quad & A_{set}(a_1 \ldots a_k) \quad \forall a_i \in A_{j-1} \\
\bigcup \quad & A_{list}(a_1 \ldots a_k) \quad \forall a_i \in A_{j-1} \\
& (\forall j > 0) \\
& \text{(Definition T7)}
\end{aligned}
$$

Finally, the set A of auths is the union of all of the A_j.

Observe that every member $a \in A$ is indeed a member of $2^{\mathbf{E}}$. I show this inductively. Clearly A_{null} and A_* are members of the power set of **E**. By rule $\mathbf{E} := \mathbf{B}$, every singleton set A_{bs} belongs to the power set of **E**. This shows the base case, $A_0 \in 2^{\mathbf{E}}$.

An auth introduced at any set A_j is either an A_{set} or an A_{list}. If it is an A_{set}, it is formed of the union of other a_i from A_{j-1}. By the induction hypothesis, each a_i is a subset of **E**, and hence their union is as well. If the new auth is instead defined by A_{list}, it is composed of strings $\llparenthesis\, x_1 \sqcup \ldots \sqcup x_n \,\rrparenthesis$. Each x_i belongs to **E**, either by its requirement to belong to a subset a_i of **E**, or by its requirement to belong to **E** itself. By rule $\mathbf{E} := \llparenthesis\, \mathbf{L} \,\rrparenthesis$, we know that **E** includes any list formed of other members of **E**, which ensures that every such string $\llparenthesis\, x_1 \sqcup \ldots \sqcup x_n \,\rrparenthesis$ is indeed in **E**.

8.4 Closure of auths under intersection

The proof that A is closed under intersection is constructive, and in fact leads directly to a concrete implementation of tag intersection. Put another way, this proof provides direct intuition why the SPKI tag intersection procedure is meaningful.

Given any two members $a_x, a_y \in A$, we wish to exhibit $a_z = a_x \cap a_y$, with $a_z \in A$. We know that there exist i_x and i_y such that $a_x \in A_{i_x}$ and $a_y \in A_{i_y}$; let us assume we have the smallest such i_x and i_y. We will show by induction over $\max(i_x, i_y)$ that $a_x \cap a_y \in A$; that is, there exists some positive integer j such that $a_x \cap a_y \in A_j$.

The base case of the induction has $\max(i_x, i_y) = 0$; that is, $a_x \in A_0$ and $a_y \in A_0$. We show in cases I, II, and III that $a_x \cap a_y \in A_0$; all possibilities for the base case appear in the italicized upper-left-hand corner of Figure 5.

The induction hypothesis assumes for all $i_x, i_y < n$, there exists a j' such that $a_x \cap a_y \in A_{j'}$. Our task, given $a_x \in A_{i_x}$ and $a_y \in A_{i_y}$ with $i_x, i_y \leq n$, is to exhibit j such that $a_x \cap a_y \in A_j$. We do so by constructing a set equal to the intersection using one of the five auth formulas A_{null}, A_*, $A_{bs}(b)$, A_{set}, or A_{list}. Then we demonstrate that the set given by the formula is in some A_j. The choice of formula depends on how a_x and a_y came to belong to A_{i_x} and A_{i_y}. For example, if i_x (the smallest index for which $a_x \in A_{i_x}$) is zero, then we know either $a_x = A_{null}$, $a_x = A_*$, or $a_x = A_{bs}(b)$ for some bytestring b. Otherwise, if $i_x > 0$, then $a_x = A_{list}(\ldots)$ or $a_x = A_{set}(\ldots)$. The same options are possible for a_y. To construct the intersection, we must consider all of the pairwise possibilities. Figure 5 maps each possibility to a proof case below. Notice we reuse base cases I and II in the inductive step.

Case I. Either $a_x = A_{null} = \varnothing$ or $a_y = A_{null} = \varnothing$, so their intersection is empty, and can be represented by A_{null}. $A_{null} = \varnothing$ belongs to A_0.

Case II. Assume without loss of generality (WOLOG) that $a_x = A_* = \mathbf{E}$ (if instead it is $a_y = A_*$, the proof works symmetrically). Then $a_x \cap a_y = a_y$. Since $a_y \in A_i$, we have $a_x \cap a_y \in A_i$.

Case III. Both a_x and a_y are singleton bytestrings. If they contain the same bytestring b, then their intersection is clearly $a_x \cap a_y = A_{bs}(b) = a_x$, which we know to be in A_i. Otherwise, the bytestrings are different, the singleton sets intersect to $\varnothing$, and we have $a_x \cap a_y = \varnothing = A_{null} \in A_0$.

Case IV. Assume WOLOG that $a_x = A_{list}(\ldots)$ and $a_y = A_{bs}(b)$. Then every member of a_x is a string beginning with the special list delimiter $\lbag$, but the single member of a_y does not. Therefore their intersection is null, a member of A_0.

Case V. Let $a_x = A_{list}(c_1 \ldots c_j)$, and $a_y = A_{list}(d_1 \ldots d_k)$. Assume WOLOG $j \leq k$. By Definition T6, the intersection $a_x \cap a_y =$

$$\left\{ \begin{array}{l} \lbag\, x_1 \text{\textvisiblespace} x_2 \text{\textvisiblespace} \ldots \text{\textvisiblespace} x_k \text{\textvisiblespace} \cdots \text{\textvisiblespace} x_n \,\rbag \\ \text{such that} \\ \quad x_i \in c_i \quad \forall i, 1 \leq i \leq j \\ \quad x_i \in \mathbf{E} \quad \forall i, j < i \\ \quad x_i \in d_i \quad \forall i, 1 \leq i \leq k \\ \quad x_i \in \mathbf{E} \quad \forall i, k < i \end{array} \right\}$$

We can rewrite the conditions as:

$$\left\{ \begin{array}{l} \lbag\, x_1 \text{\textvisiblespace} x_2 \text{\textvisiblespace} \ldots \text{\textvisiblespace} x_k \text{\textvisiblespace} \cdots \text{\textvisiblespace} x_n \,\rbag \\ \text{such that} \\ x_i \in c_i \cap d_i \quad \forall i, 1 \leq i \leq j \\ x_i \in \mathbf{E} \cap d_i \quad \forall i, j < i \leq k \\ x_i \in \mathbf{E} \cap \mathbf{E} \quad \forall i, k < i \end{array} \right\}$$

Since $a_x, a_y \in A_i$, we know by Definition T7 that $c_1 \ldots c_j \in A_{i-1}$ and $d_1 \ldots d_k \in A_{i-1}$. Let

$$e_i = \left\{ \begin{array}{ll} c_i \cap d_i & \forall i \leq j \\ d_i & \forall j < i \leq k \end{array} \right.$$

The induction hypothesis gives us j' such that $e_i \in A_{j'}$. I can now write the intersection as

$$\begin{aligned} a_x \cap a_y &= A_{list}(e_1 \ldots e_k) \\ &= \left\{ \begin{array}{l} \lbag\, x_1 \text{\textvisiblespace} x_2 \text{\textvisiblespace} \ldots \text{\textvisiblespace} x_k \text{\textvisiblespace} \cdots \text{\textvisiblespace} x_n \,\rbag \\ \text{such that} \\ \quad x_i \in e_i \quad \forall i, 1 \leq i \leq k \\ \quad x_i \in \mathbf{E} \quad \forall i, k < i \end{array} \right\} \end{aligned}$$

Because $e_i \in A_{j'}$, we conclude that $a_x \cap a_y \in A_{j'+1}$.

Case VI. Assume WOLOG $a_x = A_{set}(a_1 \ldots a_k)$. Let $a_z = A_{set}(a_1 \cap a_y, a_2 \cap a_y, \ldots a_k \cap a_y)$. We know $a_i \in A_{x_i - 1}$ for $i \leq k$, since $a_x \in A_{x_i}$. By the induction hypothesis we know that there exist $j_1 \ldots j_k$ such that $a_m \cap a_y \in A_{j_m}$ for $m \leq k$. Let $j = \max_m(j_m) + 1$. Because A_{j-1} contains every A_i for $i \leq j - 1$, we have $a_m \cap a_y \in A_{j-1}$ for all $m \leq k$. By our construction of a_z, $a_z \in A_j$.

a_y	a_x: A_{null}	A_*	$A_{bs}(b)$	$A_{list}(\ldots)$	$A_{set}(\ldots)$
A_{null}	***I***	***I***	***I***	I	I
A_*	***I***	***II***	***II***	II	II
$A_{bs}(b)$	***I***	***II***	***III***	IV	VI
$A_{list}(\ldots)$	I	II	IV	V	VI
$A_{set}(\ldots)$	I	II	VI	VI	VI

Figure 5: *Pairwise possibilities for set intersection. Roman numerals indicate the proof section that handles the given case. The emphasized entries in the upper-left corner are the cases handled in the base case of the inductive proof.*

Having covered every possible combination of a_x and a_y, we have shown the induction, and hence that A is closed under intersection. Furthermore, the cases above direct our implementation of tag intersection: given any two tags, we know which constructor (such as A_{list}) was used to create it, since tags are represented as such constructions. We can immediately apply the techniques in the preceding cases to discover a tag construct that represents the intersection of the input tags.

8.5 Tags

Tags in SPKI are approximately isomorphic with auths. There are three caveats, related to null tags, a special requirement on lists, and the special range and prefix tags.

8.5.1 The null tag

The first caveat is that SPKI has no representation for the null tag (A_{null}). The result is that the SPKI documentation must tread clumsily around the issue by saying that two authorizations "fail to intersect," rather than intersecting to a null set. By including the null set, we promote "failure" to a first-class object representable in the system.

8.5.2 Lists have an initial bytestring element

The second caveat is that lists in SPKI tags must always have at least one element, and the first element can only be a non-empty bytestring. One can readily redefine $\mathbf{B}$, $\mathbf{E}$ and A_{list} to satisfy this constraint. The basic structure of A does not change; I depend only on each A_j's membership in $2^{\mathbf{E}}$.

8.5.3 Special tags cause havoc

The third caveat is that SPKI has special tags `range` and `prefix` that define infinite subsets of the set of bytestrings. Our auth structure A can be readily extended by `prefix`, but with `range` present, A is no longer closed under intersection. Consider for example the SPKI tags:

```
(tag (* range numeric ge 0.5 le
  0.5))
(tag (* prefix 000))
```

Their intersection is $A_{range}(0.5, \leq, \leq, 0.5) \cap A_{prefix}(000)$, which we know belongs to $\mathbf{E}$. We can see, however, that it does not belong to A. The set contains an infinite number of bytestrings. We cannot construct it with an A_{prefix}, or we would end up with numeric values other than 0.5; we cannot construct it with an A_{range} or we would have prefixes other than 000. The only other way to introduce bytestrings is A_{bs}, which introduces only one at a time. We may union together any finite number of bytestrings with each application of A_{set}, but by no A_j will we have constructed the infinite set of bytestrings necessary to describe the intersection of the tags in the example.

Indeed, the trouble is concentrated in the `range` form. The intersection of two ranges with different ordering specifications can be an infinite set of bytestrings not representable with the `range` or `prefix` form.

How can we escape the dilemma? We can omit the `range` special form, but that would not provide a satisfying model of SPKI. We could introduce an intersection operator analogous to the union operator A_{set}, but that would be a hack. Since A is otherwise closed under intersection, an intersection operator should never be used except when intersecting these curious bytestring expressions, for it would only lead to needlessly larger representations for auths. Finally, we may accept the incompleteness of tags. Assume $t_3 = t_1 \cap t_2$, that is, the tag-intersection procedure run on tags t_1 and t_2 produces tag t_3. Let $A(t)$ be a function mapping a tag to the auth it represents, a typically-infinite subset of $\mathbf{E}$. If tags are complete, then $A(t_3) = A(t_1) \cap A(t_2)$. If we must sacrifice the completeness of tags, we still know that $A(t_3) \subseteq A(t_1) \cap A(t_2)$. This provides assurance that the authorization procedure is at least still sound: we will not conclude a chained delegation confers powers that are not conferred by both members of the chain.

Treating the intersection of a `range` with a `range` or `prefix` as null should not be terribly limiting in practice. When either form is used, it is specifying a value for some field with a particular interpretation; it is likely that in a real system any given field would only have one meaningful mode of comparison.

8.5.4 Semantics of special tags

SPKI contains several special forms for tags: `(*)` represents the auth A_*. `(* set ...)` represents the auth $A_{set}(...)$. `(* prefix ...)` and `(* range ...)` represent (possibly infinite) sets of bytestrings. To model these tags, we need to extend the base definition of A:

$$A_{\mathit{prefix}}(p) = \{b \mid b = ps, s \in \Sigma\} \quad \text{(Definition T8)}$$

$$A_{\mathit{range}}(f) = \{b \mid f(b) = \mathit{true}\} \quad \text{(Definition T9)}$$

$$\begin{aligned} A_0 = \quad & A_{\mathit{null}} \\ \bigcup \quad & A_* \\ \textstyle\bigcup_{b \in \mathbf{B}} \quad & A_{bs}(b) \\ \textstyle\bigcup_{p \in \mathbf{B}} \quad & A_{\mathit{prefix}}(p) \\ \textstyle\bigcup_{b \in f} \quad & A_{bs}(b) \end{aligned} \quad \text{(Definition T10)}$$

The function $f : \Sigma^* \rightarrow \{\mathit{true}, \mathit{false}\}$ selects a range of bytestrings, and is used here as an abbreviation to hide that complexity. The function depends on the specified ordering (alpha, numeric, time, binary, or date), the (optional) low and high bounds, and bits specifying whether each bound is exclusive or inclusive.

The matrix of intersection cases must now be extended to support the new possibilities (see Figure 6). This extension of course will no longer show the completeness of A under intersection (unless A_{range} is removed). But it is still useful as a thorough guide to intersecting tags.

Case VII. In this case, assume $a_x = A_{bs}(b) = \{b\}$ and $a_y = A_{\mathit{prefix}}(p)$ or $a_y = A_{\mathit{range}}(f)$. This case is very similar to case III: if $b \in a_y$, then the intersection is $a_z = A_{bs}(b) = a_x$; otherwise it is $\varnothing = A_{\mathit{null}}$.

Case VIII. We have $a_x = A_{\mathit{prefix}}(p_x)$ and $a_y = A_{\mathit{prefix}}(p_y)$. Assume WOLOG $|p_x| > |p_y|$ (p_x is a longer string). If p_y is a prefix of p_x (that is, $p_x = p_y s$), then $a_x \cap a_y = a_x$: if $b \in a_x$, $b = p_x s' = p_y s s'$, so $b \in a_y$. Otherwise, when p_y is not a prefix of p_x, the intersection is empty: $b \in a_y$ implies $b = p_y s$, and we know p_y disagrees at some symbol position with p_x, so $b \notin a_x$.

Case IX. Set a_x is a range and set a_y is a range or a prefix. Often we will treat the intersection as null (to preserve the soundness of auths). In a specific circumstance, when both a_x and a_y are ranges specified with the same ordering function, we can readily construct a range equal to the intersection by taking the more restrictive of the bounds from each input range.

Case X. Assume WOLOG a_x is a list and a_y is a range or prefix. In this case, every string in a_x begins with the list delimiter λ , and every

a_y \ a_x	A_{null}	A_*	$A_{bs}(b)$	A_{prefix}	A_{range}	$A_{list}(\ldots)$	$A_{set}(\ldots)$
A_{null}	I	I	I	***I***	***I***	I	I
A_*	I	II	II	***II***	***II***	II	II
$A_{bs}(b)$	I	II	III	***VII***	***VII***	IV	VI
A_{prefix}	***I***	***II***	***VII***	***VIII***	***IX***	***X***	***VI***
A_{range}	***I***	***II***	***VII***	***IX***	***IX***	***X***	***VI***
$A_{list}(\ldots)$	I	II	IV	***X***	***X***	V	VI
$A_{set}(\ldots)$	I	II	VI	***VI***	***VI***	VI	VI

Figure 6: *Pairwise possibilities for set intersection in the presence of the range and prefix auth constructors. The emphasized entries are additions beyond Figure 5.*

string in a_y begins with a symbol in the octet alphabet (Σ), so the intersection is null.

Notice that only case IX spoils the completeness property; striking A_{range} from our definition removes its row and column from the matrix, eliminating any reference to case IX.

8.6 The meaning of intersection

The SPKI documentation describes the intersection of two authorization tags as having two possible outcomes: a new tag or a failure to intersect. These results are meant to be interpreted differently depending on whether the intersection operation was between two delegations, or between a delegation and a specific request.

In the former case, the desire is that the tag that is the product of the intersection represents no more power than either argument tag delegated by itself, and that if the intersection fails, then the combination of the delegations is worthless.

In the latter case, the desired interpretation is that should the intersection succeed at all, the request must be authorized by the delegation tag. This interpretation makes sense if every request is more specific than any delegated permission; the only intersection possible is to return the request tag.

Our semantics lends a very concrete interpretation in both cases. When intersecting delegated permissions, it returns exactly the subset of powers granted by both input tags (modulo the incompleteness introduced by the `range` form, in which case it returns a subset of the intersection of the powers). If the tags have a null intersection, we treat that object just like any other; however, because it is an empty set of powers, it is "worthless" in the sense that no request will be authorized by it.

When authorizing requests, we intersect the delegation tag with the request tag, *and test whether the result equals the request tag.* If so, we can conclude that all of the (typically infinite) set of powers demanded by the request are granted by the delegation. If not, we conclude that some smaller (possibly empty) set of powers were granted; in any case, they are not sufficient to justify granting the request. With our semantics, it is not necessary for all requests to be finer grained than all delegated permissions. This property ensures that user-defined tag structures can be readily extended without confusing the meaning of the request authorization test.

Set containment provides a concrete, mathematically sound interpretation for specifying authorizations in an infinitely-extensible fashion.

8.7 Order dependence

The semantics of SPKI tags specifically depend on the order of elements in a list; intersection of two lists involves pairwise intersection of each list's elements. Because lists are implicitly followed by an arbitrarily-long supply of A_*s, lists are extensible in that a new property can be defined for the list and assigned to the next unused position in the list.

For example, imagine that one is defining a tag format for delegating access to a database of employee records. A first-cut tag format might look like:

```
(employee
    (id (*))
    (salary (*))
)
```

Such a definition gives users of the system the ability to delegate to others rights such as the right to inspect only employee records with a specific ID number `(employee (id 01247) (*))`, or those of employees earning more than $50,000 `(employee (*) (salary (range gt 50000)))`. Because tags are extensible, one may later decide that the ability to select employees based on anniversary year is useful, so the definition is extended to:

```
(employee
    (id (*))
    (salary (*))
    (anniversary (*))
)
```

Because list tags are followed by implicit `(*)` members, all existing delegation tags continue to be meaningful even when the new format is deployed.

What happens, however, if two independent organizations want to extend the format independently? In our example, perhaps one department of the corporation wishes to add the anniversary extension (and does so for their internal applications), and another department adds an extension representing hair color to the tag format used in their applications. The resulting extensions do not necessarily compromise security (since each sublist is annotated with the name of the attribute it refers to), but the extensions may never be used together: that third spot in the `employee` list can only contain one sublist.

There is an attractive solution. The semantic definition of auths does not preclude assigning elements to locations in lists with arbitrarily high indices. Therefore, when assigning a new extension such as `anniversary`, we could simply assign it to the index of the `employee` list given by the ordinal value of the bytestring "`anniversary`." This approach, however, is not desirable with the current definition of tag representations. A tag using the `anniversary` extension, for example, would contain some 1.2×10^{26} instances of `(*)` to place the `(anniversary ...)` sublist in the correct location.

To make the solution work, we propose a simple extension to SPKI's special-form tags, the named-attribute (`named`) form:

(* named (*attribute-name* ...))

An named-attribute tag expression always has a single list argument. The first element of the list is a bytestring (a requirement in SPKI), which we call the *attribute-name*, and the remainder of the list is the associated *value*. Let $\mathrm{ord}(b)$ be the ordinal value of a bytestring b. A list containing a `(* named (`*attribute* `...))` special form would represent the list in which the argument of the special form appears at position ord(`attribute`) in the list.

8.7.1 Handling non-bytestring attribute names

Our general semantics does not require lists to begin with a bytestring. We can easily define an ordering over Σ' rather than just Σ, and compute the location of the attribute in the parent list based on the ordinal value of the attribute name, even when that "name" is itself a list. This fix does not handle lists containing sets. The use of sets as attribute names would require some canonical ordering of the members in the set. Their use would also require a unique representation for any auth containing a set. It turns out that such a representation is indeed possible, formed by bubbling every set operator out of the inside of lists and joining them. This canonical form makes small tags into large ones; for example, `(a (* set b c d) e)` becomes `(* set (a b e) (a c e) (a d e))`. Fortunately, the $ord()$ function is only a theoretical construct used in the semantics; it would never be needed in any implementation of auth intersection.

8.7.2 Interference between ordered and named attributes

With the definition given above, one might cause an unexpected interaction between named attributes and attributes specified by their order when their positions in the list coincide. For example, if ord(A) = 65, one might construct a list with sixty-five attributes specified in order, as well as a named attribute `(* named (A cat))`. The named attribute and the sixty-fifth ordered attribute would coincide, and so far we have given no specification for how to map a list tag to an A_{list} when the tag specifies multiple auths for the same position in the list.

The semantic solution is simple: let the n^{th} ordered attribute appear at location $2n - 1$ in the list, and let each named attributed appear at location $2 \cdot \text{ord}(\textit{attribute-name})$ in the list. There are an infinite supply of odd and even list locations, and they do not interfere with one another.

What should an implementation do with a tag that specifies the same named attribute twice? It seems natural that the list location should contain the intersection of the associated values.

8.7.3 Intersection of lists containing named attributes

In any real implementation, of course, we cannot expand a list containing named attributes into its semantic form, since the length of the list grows exponentially with the length of the names of the attributes. We expect the lists to be sparse, so a sparse representation of the lists should work well. Store explicit position indices (*attribute-names* where defined, and the list index otherwise) alongside the corresponding values, with the entire collection sorted by position. The intersection routine walks the lists simultaneously and invokes itself recursively whenever it encounters two values with the same *attribute-name* or position index. As in the basic list intersection routine, if only one list specifies a value for a given position, the other list's value is assumed to be A_*, and the intersection is the explicitly specified value.

8.7.4 Recommendations for the use of ordered and named attributes

In the SPKI RFC, the authors suggest that while lists can be used to name attributes, one can omit the names for compactness. In their example,

```
(ftp (host ftp.clark.net) (dir
  /pub/cme))
```

becomes:

```
(ftp ftp.clark.net /pub/cme)
```

The rationale is that attributes are position-dependent, so there is no ambiguity when the attribute names (`host` and `dir`) are dropped.

Indeed, any correct mechanical implementations can infer the meaning of the values by their position in the list. It is likely, however, that a human implementor may incorrectly infer the intent of the values, perhaps because he only has access to example attribute values but not the names.

We recommend that attributes be supplied with names whenever possible. Whether attribute positions are specified by order or by a named-attribute special form is immaterial; that decision is one of expediency, and can be made based on the likelihood that a given attribute will be omitted from a tag specification. Providing names that document the meanings of values, however, helps avoid ambiguity, especially in a structure that is intended to be extensible in the future and by unknown parties. Our recommendation is an example of principle 1 from [AN96]: "Every message should say what it means."

8.8 Analogy with Dedekind cuts

This perspective on SPKI auth tags has a pleasant analogy to Dedekind's construction of the real numbers [BM91, pp. 15–17]. Each real number α is defined by an infinite set of rational numbers less than α; the result is continuity. The rationals are totally ordered. Every Dedekind cut respects that ordering by containing every rational less than any rational that appears in the cut.

In the construction of auths as sets of powers presented here, powers can be partially ordered, and auths respect that ordering by containing every power less than any power that appears in the auth. The result is a kind of density corresponding to the continuity of the reals. Any two unequal reals have another real between them; by analogy, any list auth can be arbitrarily subdivided into smaller auths along an arbitrary number of dimensions. This limitless extensibility makes SPKI auth tags adaptable to changing environments.

9 Modeling SPKI

The original Calculus for Access Control is useful because its principals are general enough to model several parts of a computing system, from users to trusted servers to communications channels. To formally model SPKI with our extended calculus, we first give a construction that models the delegation-control bit.

9.1 Delegation control

The SPKI document gives the motivation for including a delegation-control bit in SPKI certificates. We disagree with the argument and fall in favor of no delegation control, and for the same reasons as described in the document: delegation control is futile, and its use tempts users to divulge their keys or install signing oracles to subvert the restriction. Such subversion not only nullifies delegation control, but forfeits the benefits of auditability provided by requiring proofs of authorization. In spite of our opinion, we present a construction that models delegation control.

To model the delegation-control feature we wish to split the **says** modality into two separate modalities: "utterance," which represents a principal actually making a statement, and is never automatically inherited by other principals, and "belief," which is inherited transitively just as **says** is. Not only is introducing a new logical modality clumsy, but it would require us to support a dubious axiom, undermining the simplicity of the semantics.

Instead, we resort to an equivalent construct: we split each "real" principal A we wish to model into subprincipals A_u and A_b. A_u shall say only the things that A utters (statements that are actually signed by A's key; recall that all certificate-issuing principals in SPKI are keys), and A_b shall say all of the things that A believes. A may inherit her beliefs from other principals (because she has delegated to other subjects the authority to speak on her behalf), and furthermore A should believe anything she utters. This last condition replaces the clumsy axiom we wished to avoid; instead we enforce it by explicitly assuming the following statement for all principals A and statements s:

$$\vdash A_u \,\mathbf{says}\, s \supset A_b \,\mathbf{says}\, s \qquad \text{(Assumption E22)}$$

Certificates issued by A are statements uttered by A asserting things that A believes, so we model them as statements about A_b said by A_u. The desirable outcome is that no principal can delegate authority to make herself utter something (make A_u say something); she may only utter the statement directly (by signing it with her key).

9.2 Restriction

Recall that a SPKI 5-tuple includes five fields: issuer, subject, delegation-control bit, authorization, and validity dates. Let I and S represent the issuer and subject principals. Let T_A represent the set of primitive permissions represented by the authorization S-expression, and T_V the set of primitive permissions limited by the validity dates (assuming the effective-time encoding of Section 6.2.3). The 5-tuple can be represented this way if its delegation-control bit is set:

$$I_u \,\mathbf{says}\, S_b \stackrel{T_A \cap T_V}{\Rightarrow} I_b$$

or this way if not:

$$I_u \,\mathbf{says}\, S_u \stackrel{T_A \cap T_V}{\Rightarrow} I_b$$

A 4-tuple has a name field (N) and no authorization field or delegation-control bit. It would be encoded:

$$I_u \,\mathbf{says}\, S_b \stackrel{T_V}{\Rightarrow} I_b \cdot N$$

It seems natural that a delegation bit is meaningless for a name binding, for in SPKI, a name principal can never utter a statement directly, only a key principal can. It surprised us, however, that SPKI name-binding certificates omit the authorization field. Why not allow a principal to say the following?

$$I_u\,\textbf{says}\,(S_{2b} \overset{\{shampoo\}}{\Rightarrow} I_b \cdot N_{\mathrm{barber}})$$

As it turns out, our semantics does not support such restricted name bindings (see Section **??**).

9.3 Linked local namespaces

The subject principals in the keys above may be either keys (each directly represented by a primitive principal) or a string of names grounded in a key. Hence namespaces are "local" in that names are meaningless except relative to a globally unambiguous key; namespaces are "linked" in that the naming operation may be repeated: If $K_1 \cdot N_1$ resolves to K_2, then $K_1 \cdot N_1 \cdot N_2$ is the same as $K_2 \cdot N_2$, perhaps defined as some K_3.

We gave a logic and semantics for linked local namespaces in Section 7. We model the SPKI name subject "george: (name fred sam)" with the principal expression $K_{\mathrm{george}} \cdot N_{\text{“fred”}} \cdot N_{\text{“sam”}}$. Substituting the principal expression for S_b, a 4-tuple takes on the general appearance:

$$I_u\,\textbf{says}\,((K_S \cdot N_1 \cdots N_k) \overset{T_V}{\Rightarrow} I_b \cdot N_0)$$

9.4 Threshold subjects

A threshold subject is a group of n principals who are authorized by a certificate only when k of the principals agree to the requested action. Such certificates are really just an abbreviation for a combinatorially-long $\binom{n}{k}$ list of conjunction statements. For example, a certificate with a 2-of-3 threshold subject naming principals P_1, P_2, and P_3 and an issuer A can be represented as:

$$P_1 \wedge P_2 \Rightarrow A$$
$$P_1 \wedge P_3 \Rightarrow A$$
$$P_2 \wedge P_3 \Rightarrow A$$

Hence the logic easily captures threshold subjects, although any tractable implementation would obviously want to work with them in their unexpanded form.

9.5 Auth tags

The "auth tags" used in authorization fields in SPKI represent sets of primitive statements. Therefore, we simply model them using mathematical sets.

9.6 Tuple reduction

The SPKI access-control decision procedure is called "tuple reduction." A request is granted if it can be shown that a collection of certificates reduce to authorize the request. The reduced tuple's subject must be the key that signed the request; the tuple's issuer must represent the server providing the requested service; and the specific request must belong to the authorization tag of the reduced tuple.

It is clear that tuple reduction is sound with respect to our extended logic. When 5- and 4-tuples are encoded in the logic as shown in Sections 7 and 9.2, tuple-reduction simply constructs a proof from several applications of Theorem E6 and Axiom E17.

9.7 Validity conditions

An optional validity condition, such as a certificate revocation list, a timed revalidation list, or a one-time validation, can be encoded in the logic using a conjunction. For example, a certificate requiring a timed revalidation would be interpreted

$$A\,\textbf{says}\,(B \wedge (R|H_1)) \Rightarrow A$$

to mean that principal R must verify that this certificate (with hash H_1) is valid. Principal R signs a revalidation instrument I with a short validity interval T_V

$$R\,\textbf{says}\,I \overset{T_V}{\Rightarrow} R$$

and a given revalidation instrument would agree with all valid outstanding certificates:

$$I \,\textbf{says}\, \mathbf{0} \Rightarrow I|H_1$$
$$I \,\textbf{says}\, \mathbf{0} \Rightarrow I|H_2$$
$$\vdots$$

The principal $\mathbf{0}$ has relation $\mathcal{R}(\mathbf{0}) = \varnothing$, so that every principal speaks for $\mathbf{0}$. Using the logic, we can reason that

$$\mathbf{0} \Rightarrow I|H_1 \stackrel{T_V}{\Rightarrow} R|H_1$$

and since $B = B \wedge \mathbf{0}$, $B \stackrel{T_V}{\Rightarrow} A$. Notice the treatment of a certificate's hash as a principal. In the logic, principals are general entities and can be used to represent many objects and actors.

Negative certificate revocation lists can be handled similarly; an implementation examining a revocation list would conclude $I \,\textbf{says}\, \mathbf{0} \Rightarrow I|H_1$ for any H_1 *not* present in the list.

One-time revalidations are meant to be interpreted as having a zero validity interval. A system verifying a request s creates a nonce E, understanding $E \,\textbf{says}\, s$, and sends it to the revalidator R. R replies with a statement meant to be interpreted

$$R \,\textbf{says}\, E \stackrel{\{s\}}{\Rightarrow} R|H_1$$

Now both B_1 and $E \stackrel{\{s\}}{\Rightarrow} R|H_1$ say s, so $A \,\textbf{says}\, s$. Any future request of the same sort will require another revalidation, for its s will have a different effective time.

9.8 Safe extensions

Our semantics suggests that SPKI may be safely extended to support a variety of principals other than public keys. Channels protected by secret keys or a trusted computing base, for example, are easily modeled as principals in the logic. Conjunct principals $(A \wedge B)$ are not first-class entities in SPKI, although they can appear as threshold subjects; an extended SPKI might exploit Theorem E20.

Quoting principals are also missing from SPKI; Lampson *et al.* give nice examples showing how quoting can help a multiplexed server or communications channel differentiate when it is working on behalf of one client versus another [LABW92, Sections 4.3, 6.1, 6.2, and 7.1]. Without quoting, such a server has permission to make statements for either client, so it must perform an access-control check in advance of relaying a client's statement. Quoting lets the multiplexed server defer the complete access-control decision to the final resource server that verifies the proof. The result is a smaller trusted computing base and improved auditability.

9.9 Dangerous extensions

In this section, we argue that SPKI auth tags should not be extended to represent logical negations. If $\mathcal{B}$ speaks for $\mathcal{A}$ regarding multiple restriction sets, the semantics suggest that $\mathcal{B}$ actually has some authority not explicitly mentioned in either set. For example,

$$(\mathcal{B} \stackrel{\{\sigma,\tau\}}{\rightarrow} \mathcal{A}) \supset (\mathcal{B} \stackrel{\{\sigma\wedge\tau\}}{\rightarrow} \mathcal{A}) \qquad \text{(Axiom E23)}$$

means that a principal believed on a set of statements is also believed on their conjuncts. This conclusion seems fairly natural, but it is interesting to note that a restriction set actually permits more statements than it represents explicitly.

With our projected version of Abadi's speaks-for semantics, not only does

$$(\mathcal{B} \stackrel{\{\sigma,\tau\}}{\Rightarrow} \mathcal{A}) \supset (\mathcal{B} \stackrel{\{\sigma\wedge\tau\}}{\Rightarrow} \mathcal{A}) \qquad \text{(Axiom E24)}$$

hold, but also:

$$(\mathcal{B} \stackrel{\{\sigma\}}{\Rightarrow} \mathcal{A}) \supset (\mathcal{B} \stackrel{\{\neg\sigma\}}{\Rightarrow} \mathcal{A}) \qquad \text{(Axiom E25)}$$

This result implies that given authority on a set of primitive statements, a principal also has authority on any propositional formula constructed from those statements. It is surprising, for even if only $B \stackrel{\{s\}}{\Rightarrow} A$ is explicitly granted, B can also cause A to say the negation of s.

Perhaps scarier still is that

$$\begin{aligned} B \stackrel{\{\sigma\}}{\Rightarrow} A \supset B &\stackrel{\{\sigma,\neg\sigma\}}{\Rightarrow} A \\ &\supset (B \,\textbf{says}\, \text{false}) \supset (A \,\textbf{says}\, \text{false}) \end{aligned}$$

The conclusion is the definition of Abadi's $\mapsto$ relation:

> "Intuitively, $A \mapsto B$ means that there is something that A can do (say *false*) that yields an arbitrarily strong statement by B (in fact, *false*). Thus, $A \mapsto B$ means that A is at least as powerful as B in practice." [ABLP93, p. 713]

With these semantics, one might fear that no restriction is actually meaningful. How might we escape it? One option is to abandon the **K** axiom ($A\,\mathbf{believes}\,s \wedge A\,\mathbf{believes}\,(s \supset t) \supset A\,\mathbf{believes}\,t$), so that principals no longer believe every consequence of their beliefs. This option is undesirable because it cripples the logic to only operate outside the scope of belief operators.

A second option is to both disallow negative statements in restriction sets and to use the weaker $B \xrightarrow{T} A$ relation instead of $B \overset{T}{\Rightarrow} A$ to model delegation.

A third option is to prevent principals from making contradictory statements. This is difficult in general in a distributed system. One approach is to prevent principals from making negative statements at all. SPKI takes this approach. Its tags, which represent both restriction sets and individual statements, cannot represent both a statement and its logical negation. We provide a formal treatment of tags in Section 8.

Another extension might be to allow SPKI name bindings (4-tuples) to include authorization restrictions. As mentioned in Section **??**, the semantics does not support this seemingly-natural extension.

We conclude that in certain dimensions, SPKI is as strong as it can be. Changing SPKI by allowing principals to make negative statements or by allowing negative statements in restriction sets would push SPKI "over the edge," making its restrictions meaningless. Those proposing to augment SPKI or other systems based on a logic such as that presented here must be wary of this hazard.

10 Summary

We extend the Calculus for Access Control and its underlying possible-worlds semantics to support restricted delegation, delegation control, and local namespaces. To define the semantics of restricted delegation, we project a model to a set of worlds distinguished only by statements in the restriction set. The resulting system provides intuition and a formal framework in which we reason about the current SPKI system and possible extensions to SPKI.

One of the advantages our formal framework is that it represents the many complicated features of SPKI with three simple concepts: *principal*, *statement*, and *name*. Features such as threshold subjects and on-line validations can be modeled with compound principals and idiomatic statements. The simplicity also suggests that SPKI may be safely integrated with systems with notions of "principal" other than SPKI's public keys; such principals are desirable because they can exploit fast local or secret-key-protected channels. We are applying our results in just this way in a prototype system currently under implementation.

Our formalism also warns of the danger of apparently-harmless extensions. In our semantics, allowing a principal to utter both a statement and its negation or allowing restricted delegation to a name binding would reduce restricted delegation to meaninglessness. It would be imprudent to so extend SPKI without developing an alternate semantics that gives the extension meaning. One might also assume that delegation over two sets of permissions should combine to represent a delegation over the union of the permissions, but Result E7 suggests that this is not the case.

Acknowledgements

Thanks to John Lamping, who patiently helped Jon understand logical proof systems and semantic models. Thanks also Jon Bredin, Valeria de Paiva, Mark Montague and Larry Gariepy for their discussions, which helped refine the idea. Thanks to the USENIX organization for funding our research on this topic.

References

[Aba98] M. Abadi. On SDSI's linked local name spaces. *Journal of Computer Security*, 6(1-2):3–21, 1998.

[ABLP93] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, September 1993.

[AN96] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

[Aur98] Tuomas Aura. On the structure of delegation networks. In *Proceedings of the Eleventh IEEE Computer Security Foundations Workshop*, pages 14–26, 1998.

[BM91] David F. Belding and Kevin J. Mitchell. *Foundations of Analysis*. Prentice-Hall, 1991.

[EFL+99] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. SPKI certificate theory, October 1999. Internet RFC 2693.

[FHMV95] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[Gol73] William Goldman. *The Princess Bride*. Ballantine, 1973.

[HC96] G. E. Hughes and M. J. Cresswell. *A New Introduction to Modal Logic*. Routledge, 1996.

[HvdM99] Joseph Y. Halpern and Ronald van der Meyden. A logic for SDSI's linked local name spaces. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 111–122, 1999.

[LABW92] Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, November 1992.

[WABL94] Edward Wobber, Martín Abadi, Michael Burrows, and Butler Lampson. Authentication in the Taos operating system. *ACM Transactions on Computer Systems*, 12(1):3–32, February 1994.

A Proofs

A.1 Construction of ϕ_T

Definition Definition E12 presupposed the existence of a projection function ϕ_T. We construct such a function now, and show that it satisfies the definition. Let $\overline{W} = 2^T$; that is, worlds in $\overline{M}$ are subsets of T. Define

$$\phi_T(w) = \overline{w} \in \overline{W}$$
$$\text{where } (\sigma \in \overline{w}) \equiv (w \in \mathcal{E}(\sigma)) \ \forall \sigma \in T \qquad \text{(Definition E26)}$$

Necessity. Given $\phi_T(w) = \overline{w} = \phi_T(w')$, we know $\forall \sigma \in T,\ \sigma \in \overline{w}$ **iff** $w \in \mathcal{E}(\sigma)$, and likewise, $\forall \sigma \in T,\ \sigma \in \overline{w}$ **iff** $w' \in \mathcal{E}(\sigma)$. Therefore $\forall s \in T,\ w \in \mathcal{E}(\sigma)$ **iff** $w' \in \mathcal{E}(\sigma)$, and we conclude $w \cong_T w'$.

Sufficiency. From the definition of $w \cong_T w'$, we know $\forall \sigma \in T,\ w \in \mathcal{E}(\sigma)$ **iff** $w' \in \mathcal{E}(\sigma)$. Let $\overline{w} = \{\sigma \in T | w \in \mathcal{E}(\sigma)\}$ and $\overline{w}' = \{\sigma \in T | w' \in \mathcal{E}(\sigma)\}$. From our hypothesis we know that the conditions on $\overline{w}$ and $\overline{w}'$ are the same, so $\phi_T(w) = \overline{w} = \overline{w}' = \phi_T(w')$.

In the following proofs, we generally use a bar ($\overline{w}$) to indicate a member of an equivalence class constructed as shown here.

A.2 Equivalence of ϕ_T^R and ϕ_T^+ definitions of $\stackrel{T}{\Rightarrow}$

We now justify our claim in Section 6.1 that Definition E14 and Definition E16 are equivalent.

Necessity. Assume $\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A}$ holds according to Definition E14:

$$\forall w_0' \ \ (\phi_T^w(\mathcal{R}(\mathcal{A})(w_0')) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w_0')))$$

For all $\langle w_0, w_1\rangle$,

$$\begin{aligned}
\langle w_0, w_1\rangle \in \mathcal{R}(\mathcal{A}) &\supset w_1 \in \mathcal{R}(\mathcal{A})(w_0)\\
&\supset\ \overline{w}_1 \in \phi^w_T(\mathcal{R}(\mathcal{A})(w_0)),\\
&\qquad \overline{w}_1 = \phi_T(w_1)\\
&\supset \overline{w}_1 \in \phi^w_T(\mathcal{R}(\mathcal{B})(w_0))\\
&\qquad\text{(using the assumption)}\\
&\supset\ \exists w'_1 \cong_T w_1,\\
&\qquad \langle w_0, w'_1\rangle \in \mathcal{R}(\mathcal{B})(w_0)\\
&\supset \langle w_0, w_1\rangle \in \phi^+_T(\mathcal{R}(\mathcal{B}))
\end{aligned}$$

Sufficiency. Assume $\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A}$ holds according to Definition E16:

$$\mathcal{R}(\mathcal{A}) \subseteq \phi^+_T(\mathcal{R}(\mathcal{B}))$$

Given w_0 and $\overline{w}_1 \in \phi^w_T(\mathcal{R}(\mathcal{A})(w_0))$, we know that there is some $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$, with $\overline{w}_1 = \phi_T(w_1)$. We rewrite the statement $\langle w_0, w_1\rangle \in \mathcal{R}(\mathcal{A})$, and invoke the assumption to get $\langle w_0, w_1\rangle \in \phi^+_T(\mathcal{R}(\mathcal{B}))$. Now we know there exists $\langle w_0, w'_1\rangle \in \mathcal{R}(\mathcal{B})$ with $w'_1 \cong_T w_1$. Changing notation again, $w'_1 \in \mathcal{R}(\mathcal{B})(w_0)$. Since $w'_1 \cong_T w_1$, we know $\overline{w}_1 = \phi_T(w'_1)$, and we may conclude $\overline{w}_1 \in \phi^w_T(\mathcal{R}(\mathcal{B})(w_0))$.

Together, the two implications show the equivalence.

A.3 An undesirable semantics for $\stackrel{T}{\Rightarrow}$

Notice that ϕ^+_T projects only the destination world of each edge in a relation. Why do we not project both ends of the relation? Such a definition actually does not preserve our most basic intuition, that $B \stackrel{T}{\Rightarrow} A \supset B \stackrel{T}{\rightarrow} A$. In the model in Figure 7, the dotted ovals depict the equivalence classes under T; projecting both ends of the edges in $\mathcal{R}(A)$ gives $\{\langle T, \varnothing\rangle\}$, as does $\mathcal{R}(B)$. From world w_0, however, B **says** s but not A **says** s.

Given a relation $\langle w_0, w_1\rangle$, then, the reason we only project w_1 is this: w_0 is affected by what statements are true at w_1; substituting other worlds equivalent with respect to T does no harm. Substituting other worlds for w_0, on the other hand, changes what statements we consider true at w_0.

A.4 Proof of soundness

In this section, we show that our extension to Lampson's calculus is still a sound axiomatization of the presented semantics. Like Lampson's original logic, ours is based on a conventional Kripke semantics of modal logic. The conventional proofs of soundness

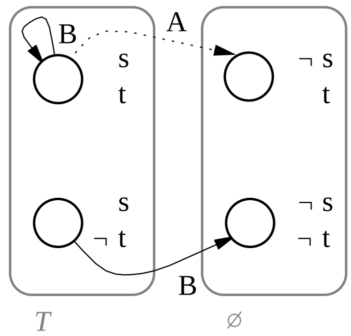


Figure 7: *In this example, $T = \{s\}$. Notice that $B \stackrel{T}{\not\rightarrow} A$.*

for Axiom S1, Rule S2, Axiom S3, and Rule S4 apply. Our extensions define $\mathcal{E}$ for a new formula ($\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A}$) and $\mathcal{R}$ for a new principal ($\mathcal{A}\cdot N$), but do not perturb Abadi's original semantics for the calculus for access control. Because those semantics do not depend on any particular structure in $\mathcal{E}$ or $\mathcal{R}$, the axioms of the calculus remain sound in our extended calculus.

Our present task is to show that the axioms of our extensions are sound.

Axiom E1. This axiom follows easily from Definition E14. For all w_0,

$$\begin{aligned}
\phi^w_T(\mathcal{R}(\mathcal{A})(w_0)) &\subseteq \phi^w_T(\mathcal{R}(\mathcal{B})(w_0))\\
&\subseteq \phi^w_T(\mathcal{R}(\mathcal{C})(w_0)) \qquad \square
\end{aligned}$$

The following lemma shows that ϕ^+_T preserves the union operation. Let R_1 and R_2 be relations.

$$\begin{aligned}
&\langle w_0, w_1\rangle \in \phi^+_T(R_1 \cup R_2)\\
&\quad\equiv \exists\, w'_1 \cong_T w_1,\ \langle w_0, w'_1\rangle \in R_1 \cup R_2\\
&\quad\equiv \exists\, w'_1 \cong_T w_1,\\
&\qquad\quad \langle w_0, w'_1\rangle \in R_1 \vee \langle w_0, w'_1\rangle \in R_2\\
&\quad\equiv \quad \exists\, w'_1 \cong_T w_1,\ \langle w_0, w'_1\rangle \in R_1\\
&\qquad \vee \exists\, w'_1 \cong_T w_1,\ \langle w_0, w'_1\rangle \in R_2\\
&\quad\equiv \langle w_0, w_1\rangle \in \phi^+_T(R_1) \vee \langle w_0, w_1\rangle \in \phi^+_T(R_2)\\
&\quad\equiv \langle w_0, w_1\rangle \in \phi^+_T(R_1) \cup \phi^+_T(R_2)
\end{aligned}$$

From this equivalence we conclude

$$\phi^+_T(R_1 \cup R_2) = \phi^+_T(R_1) \cup \phi^+_T(R_2) \qquad \text{(Lemma E27)}$$

Axiom E2. We assume the premise in terms of Definition E14:

$$\forall\, w'_0\ (\phi^w_T(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi^w_T(\mathcal{R}(\mathcal{B})(w'_0)))$$

We can readily reason for all w_0:

$$\begin{aligned}
&\phi_T^w(\mathcal{R}(\mathcal{A}\wedge\mathcal{C})(w_0))\\
&\quad=\phi_T^w((\mathcal{R}(\mathcal{A})\cup\mathcal{R}(\mathcal{C}))(w_0))\\
&\quad=\phi_T^w(\mathcal{R}(\mathcal{A})(w_0)\cup\mathcal{R}(\mathcal{C})(w_0))\\
&\quad=\phi_T^w(\mathcal{R}(\mathcal{A})(w_0))\cup\phi_T^w(\mathcal{R}(\mathcal{C})(w_0))\\
&\quad\subseteq\phi_T^w(\mathcal{R}(\mathcal{B}))\cup\phi_T^w(\mathcal{R}(\mathcal{C}))\\
&\quad=\phi_T^w(\mathcal{R}(\mathcal{B})(w_0)\cup\mathcal{R}(\mathcal{C})(w_0))\\
&\quad=\phi_T^w((\mathcal{R}(\mathcal{B})\cup\mathcal{R}(\mathcal{C}))(w_0))\\
&\quad=\phi_T^w(\mathcal{R}(\mathcal{B}\wedge\mathcal{C})(w_0)) \qquad \square
\end{aligned}$$

Axiom E3. This axiom has a symmetric consequence, so we only show the first conjunct. For all worlds w_0,

$$\begin{aligned}
\phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) &\subseteq \phi_T^R((\mathcal{R}(\mathcal{A})\cup\mathcal{R}(\mathcal{B}))(w_0)) && \text{(Lemma E27)}\\
&\subseteq \phi_T^R(\mathcal{R}(\mathcal{C})(w_0)) && \text{(premise)}
\end{aligned}$$

$\square$

We digress to point out that we may discard "identical worlds" from a model without loss of generality. That is, imagine we have a model $\mathcal{M}$ with two worlds w_1 and w_2 where $w_1\in\mathcal{E}(\sigma)$ **iff** $w_2\in\mathcal{E}(\sigma)$ for every formula $\sigma\in\Sigma^*$. The extra world w_2 appears in every $I(s)$ that w_1 appears in. Any edge in any relation ending in w_1 has a related edge ending in w_2 ($\langle w,w_1\rangle\in J(\mathcal{A})\equiv\langle w,w_2\rangle\in J(\mathcal{A})$); likewise edges starting at w_1 have a related edge starting at w_2 in every relation. The same holds for the relations in the name interpretation function $K(\mathcal{A},N)$. It is clear that the extension function $\mathcal{R}$, and hence $\mathcal{E}$, have the same overlap with respect to w_1 and w_2, so that $w_1\in\mathcal{E}(\sigma)\equiv w_2\in\mathcal{E}(\sigma)$.

Given this definition, we can build a model $\mathcal{M}'=\langle W',w_0',I',J',K'\rangle$ that discards w_2:

$$\begin{aligned}
W' &= W-\{w_2\}\\
w_0' &= \begin{cases} w_1 & \text{if } w_0\\ w_0 & \text{otherwise}\end{cases}\\
I'(s) &= I(s)-\{w_2\}\\
J'(\mathcal{A}) &= J(\mathcal{A})-\{\langle w,w'\rangle | w=w_2\vee w'=w_2\}\\
K'(\mathcal{A},N) &= K(\mathcal{A},N)\\
&\quad -\{\langle w,w'\rangle | w=w_2\vee w'=w_2\}
\end{aligned}$$

Happily, $\mathcal{M}'$ preserves every consequence of $\mathcal{M}$: $(\mathcal{M}\models\sigma)\equiv(\mathcal{M}'\models\sigma)$. Why? Whenever $w_0\in\mathcal{E}(\sigma)$, $w_0'\in\mathcal{E}'(\sigma)$, either for exactly the same reasons (when $w_0\neq w_2$), or because $w_0=w_2$, so $w_0=w_2\in\mathcal{E}(\sigma)\equiv w_1\in\mathcal{E}(\sigma)$, and then $w_0'\in\mathcal{E}'(s)$ for the same reasons that $w_1\in\mathcal{E}(s)$.

Convinced that duplicate worlds do not alter the consequences of a model, we may now assume that no models contain identical worlds, without damaging our semantics. If we know $w_1\neq w_2$, we can assume the existence of a formula σ with $(w_1\in\mathcal{E}(\sigma))\not\equiv(w_2\in\mathcal{E}(\sigma))$, and conclude that $w_1\not\cong_{\mathcal{U}} w_2$ (by Definition E11). Therefore, $\phi_{\mathcal{U}}$ is bijective:

$$w_1\neq w_2\supset\phi_{\mathcal{U}}(w_1)\neq\phi_{\mathcal{U}}(w_2)$$

By the definition of ϕ_T^+ it is obvious that any relation $R\in\phi_T^+(R)$. But when $T=\mathcal{U}$, the converse is also true:

$$\begin{aligned}
&\langle w_0,w_1\rangle\in\phi_{\mathcal{U}}^+(R)\\
&\quad\supset\exists\, w_1' \text{ such that } \langle w_0,w_1'\rangle\in R,\\
&\qquad\quad \phi_{\mathcal{U}}(w_1')=\phi_{\mathcal{U}}(w_1)\\
&\quad\supset w_1'=w_1\\
&\quad\supset\langle w_0,w_1\rangle\in R
\end{aligned}$$

Now we have $\phi_{\mathcal{U}}^+(R)=R$.

Axiom E4. Expanding the definition of $B\overset{\mathcal{U}}{\Rightarrow}A$ and applying the previous result gives $\mathcal{R}(A)\subseteq\phi_{\mathcal{U}}^+(\mathcal{R}(B))=\mathcal{R}(B)$, which satisfies the definition of $B\Rightarrow A$. $\square$

Justifying axiom Axiom E5 requires two lemmas that relate representatives of equivalence classes under different projections.

First, a representative of a projection due to a small set has a "big brother" in any projection due to a superset, and the structure of the brothers is closely related:

$$\begin{aligned}
&\overline{w}_1'\in\phi_{T'}^w(S_w),\ T'\subseteq T\\
&\supset\exists\,\overline{w}_1\in\phi_T^w(S_w),\ \overline{w}_1'=\overline{w}_1\cap T' \qquad \text{(Lemma E28)}
\end{aligned}$$

Proof. By the first premise, there is a $w_1\in S_w$ where $\overline{w}_1'=\phi_{T'}(w_1)$. From Definition E26 we know

$$(\sigma\in\overline{w}_1')\equiv(w_1\in\mathcal{E}(\sigma))\ \forall\,\sigma\in T' \qquad (1)$$

Let $\overline{w}_1=\phi_T(w_1)$; since $w_1\in S_w$, $\overline{w}_1\in\phi_T^w(S_w)$. Having exhibited $\overline{w}_1$, we need only show $\overline{w}_1\cap T=\overline{w}_1'$.

We again invoke Definition E26 to get

$$(\sigma\in\overline{w}_1)\equiv(w_1\in\mathcal{E}(\sigma))\ \forall\,\sigma\in T \qquad (2)$$

First, $\sigma\in\overline{w}_1\cap T'$ means both $\sigma\in T'$, and because $T'\subseteq T$, $\sigma\in T$. The latter allows us to use (2) to write $w_1\in\mathcal{E}(\sigma)$, and then we invoke (1) to get $\sigma\in\overline{w}_1'$. Conversely, $\sigma\in\overline{w}_1'$ means $\sigma\in T'$ and hence $\sigma\in T$. We apply (1) to get $w_1\in\mathcal{E}(\sigma)$, and apply (2) to get $\sigma\in\overline{w}_1$. Now we have shown $\overline{w}_1\cap T'=\overline{w}_1'$, proving the lemma. $\square$

The second lemma is approximately the converse of the first:

$$\overline{w}_1 \in \phi_T^w(S_w),\ \overline{w}'_1 = \overline{w}_1 \cap T'\ T' \subseteq T$$
$$\supset \overline{w}'_1 \in \phi_{T'}^w(S_w) \qquad \text{(Lemma E29)}$$

Proof. The first premise, by Definition E13, implies the existence of a $w_1 \in R$, and Definition E26 lets us write

$$(\sigma \in \overline{w}_1) \equiv (w_1 \in \mathcal{E}(\sigma))\ \forall\, \sigma \in T \qquad (1)$$

For every $\sigma \in T'$, all of the following hold:

$$\begin{aligned} \sigma &\in T && \text{(third premise)} \\ (\sigma \in \overline{w}_1) &\equiv (w_1 \in \mathcal{E}(\sigma)) && (1) \\ (\sigma \in \overline{w}_1 \cup T') &\equiv (w_1 \in \mathcal{E}(\sigma)) && \\ (\sigma \in \overline{w}'_1) &\equiv (w_1 \in \mathcal{E}(\sigma)) && \text{(second premise)} \end{aligned}$$

This last result implies that $\overline{w}'_1 = \phi_{T'}(w_1)$, which is sufficient to prove the conclusion of the lemma. □

Axiom E5. We take as our hypothesis $\mathcal{M} \models B \overset{T}{\Rightarrow} A$, that is:

$$\phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$$

Given any world w_0 and sets $T' \subseteq T$, we assume $\overline{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0))$ and set out to prove $\overline{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{B})(w_0))$. By the assumption and Lemma E28, we know

$$\exists\, \overline{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)),\ \overline{w}'_1 = \overline{w}_1 \cap T'$$

The hypothesis gives $\overline{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$, which satisfies the premise for Lemma E29. Hence we know $\overline{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{B})(w_0))$, and we have proven that

$$\forall\, w_0,\ (\phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0)) \subseteq \phi_{T'}^w(\mathcal{R}(\mathcal{B})(w_0))) \qquad □$$

Theorem E6. Apply Axiom E5 twice to the premises to get two relations restricted by $S \cup T$, then apply Axiom E1 to collapse them into the relation in the conclusion. □

Result E7. Figure 8 gives a counterexample that justifies the result. The diagram in the figure models $B \overset{S}{\Rightarrow} A$ and $B \overset{T}{\Rightarrow} A$. The statement $B \overset{S \cup T}{\Rightarrow}$, however, fails. Projecting the model under $S \cup T$ gives the original picture, since each world falls in a separate equivalence class. Notice that B says $\neg(s \wedge \neg t)$: that statement is true in both worlds B considers possible. But A does not believe it, since A can see the lower-left world, where the statement is false.

Why should this result be intuitive or desirable? Recall from Section 9.9 that the strength of $\overset{T}{\Rightarrow}$ means that a delegation regarding T may imply a delegation regarding a larger set T^* that includes formulas constructed from the members of T. In our example, B speaks for A regarding formulas composed exclusively with the primitive s or the primitive t, but not regarding formulas combining the two. The closure of the restriction set $S \cup T$ includes formulas such as $\neg(s \wedge \neg t)$.

Axiom E8. Assume the premise in terms of Definition E14:

$$\forall\, w'_0\ (\phi_T^w(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w'_0)))$$

Let $\overline{w}$ belong to $\phi_T^w(\mathcal{R}(\mathcal{C}|\mathcal{A})(w_0))$. The semantics for quoting gives $\overline{w} \in \phi_T^w((\mathcal{R}(\mathcal{C}) \circ \mathcal{R}(\mathcal{A}))(w_0))$. An edge only exists in a composition if we have w_1 and w_2 such that $\langle w_0, w_1 \rangle \in \mathcal{R}(\mathcal{C})$ and $\langle w_1, w_2 \rangle \in \mathcal{R}(\mathcal{A})$; Definition E13 guarantees that we have such w_1, w_2 with $\overline{w} = \phi_T(w_2)$.

Since $w_2 \in \mathcal{R}(\mathcal{A})(w_1)$, we can use the assumption to show the existence of $w'_2 \in \mathcal{R}(\mathcal{B})(w_1)$ with $\phi_T(w'_2) = \phi_T(w_2) = \overline{w}$. That means that $\overline{w} \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_1))$, and hence $\overline{w} \in \phi_T^w((\mathcal{R}(\mathcal{C}) \circ \mathcal{R}(\mathcal{B}))(w_0))$. By the definition of quoting, we arrive at $\overline{w} \in \phi_T^w(\mathcal{R}(\mathcal{C}|\mathcal{B})(w_0))$, which proves the conclusion. □

Result E9. The model in Figure 9 is a counterexample for $T = \{s\}$ that shows the result. Notice that $B \overset{T}{\Rightarrow} A$: $\mathcal{R}(A)$'s only edge goes from w_0 to the equivalence class of worlds where s is true, and $\mathcal{R}(B)$ also has such an edge (the loop at w_0). When we compose the relations, however, we see that $B|C$ **says** s, but not $A|C$ **says** s. The equivalence classes of $\{C$ **says** $s\}$ are different than the equivalence classes of $\{s\}$.

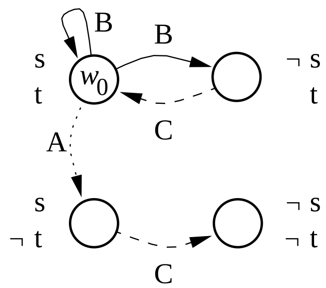
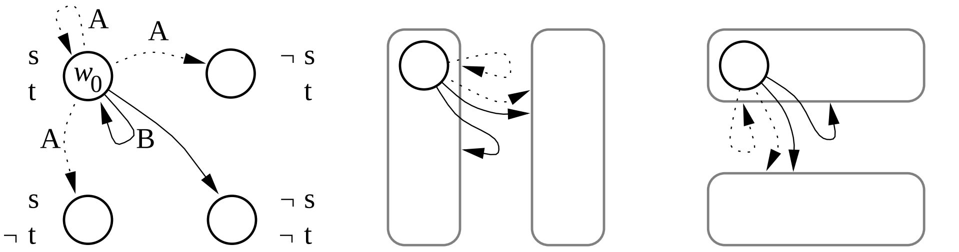


Figure 9: *A model that demonstrates Result E9.*

Axiom E10. Inductively applying Axiom E25 and Axiom E24 shows as a theorem that $B \overset{T}{\Rightarrow} A$ implies $B \overset{T^*}{\Rightarrow} A$. Therefore, we may immediately replace the premise of this axiom with $B \overset{((T^*)C)^*}{\Rightarrow} A$, which follows by the theorem from the original premise. Herein we omit the parentheses for the postfix set operators * and C, and simply write T^*C^*.



In this model, A's relation at w_0 is not a subset of B's.

Projected under $S = \{s\}$, however, the subset relation holds ...

... as it does under $T = \{t\}$.

Figure 8: *A counterexample showing why two delegations for sets S and T do not imply a delegation for set $S \cup T$ (Result E7).*

Hence we begin with the hypothesis that

$$\mathcal{R}(\mathcal{A}) \subseteq \phi^{+}_{T^*C^*}(\mathcal{R}(\mathcal{B}))$$

We are given some $w_0 \in W$ and the existence of $\overline{w}_2 \in \phi^w_T(\mathcal{R}(\mathcal{A}|\mathcal{C})(w_0))$. The set can be rewritten $\phi^w_T((\mathcal{R}(\mathcal{A}) \circ \mathcal{R}(\mathcal{C}))(w_0))$, so we know that there exist w_1 and w_2, where

$$\begin{aligned} \langle w_0, w_1 \rangle &\in \mathcal{R}(\mathcal{A}) \\ \langle w_1, w_2 \rangle &\in \mathcal{R}(\mathcal{C}) \\ \overline{w}_2 &= \phi_{T^*C^*}(w_2) \end{aligned}$$

The last expression means that for all $\sigma \in T$, $\sigma \in \overline{w}_2$ if and only if $w_1 \in \mathcal{E}(\sigma)$.

Define the formula

$$\tau_2 = \bigwedge_{\sigma \in T} \begin{cases} \sigma & \text{if } \sigma \in \overline{w}_2 \\ \neg\sigma & \text{otherwise} \end{cases}$$

Intuitively, τ_2 is true at precisely those worlds that map to $\overline{w}_2$ under ϕ_T. We have constructed τ_2 such that $w_2 \in \mathcal{E}(\tau_2)$.

Since $\langle w_1, w_2 \rangle \in \mathcal{R}(\mathcal{C})$, we know $\mathcal{R}(\mathcal{C}) \not\subseteq \mathcal{E}(\neg\tau_2)$, and therefore $w_1 \notin \mathcal{E}(\mathcal{C}\,\textbf{says}\,\neg\tau_2)$, and finally $w_1 \in \mathcal{E}(\neg\mathcal{C}\,\textbf{says}\,\neg\tau_2)$. The propositional closure of T ensures that each conjunct of τ_2, and thus τ_2 itself and $\neg\tau_2$, appear in T^*. The modal closure over "$\mathcal{C}$ **says**" ensures that $(\mathcal{C}\,\textbf{says}\,\neg\tau_2) \in T^*C$, and therefore $(\neg\mathcal{C}\,\textbf{says}\,\neg\tau_2) \in T^*C^*$.

Now we may employ the hypothesis to show that there exists a $w'_1 \in \mathcal{R}(\mathcal{B})(w_0)$ with $w'_1 \cong_{T^*C^*} w_1$. It follows that:

$$\begin{aligned} w'_1 &\in \mathcal{E}(\neg\mathcal{C}\,\textbf{says}\,\neg\tau_2) \\ &= W - \mathcal{E}(\mathcal{C}\,\textbf{says}\,\neg\tau_2) \\ &= W - \{w | \mathcal{R}(\mathcal{C})(w) \subseteq \mathcal{E}(\neg\tau_2)\} \\ &= \{w | \mathcal{R}(\mathcal{C})(w) \not\subseteq \mathcal{E}(\neg\tau_2)\} \\ &= \{w | \exists\, w'_2 \in \mathcal{R}(\mathcal{C})(w),\ w'_2 \notin \mathcal{E}(\neg\tau_2)\} \\ &= \{w | \exists\, w'_2 \in \mathcal{R}(\mathcal{C})(w),\ w'_2 \in \mathcal{E}(\tau_2)\} \end{aligned}$$

That is, we know there is a $w'_2 \in \mathcal{E}(\tau_2)$, with $\langle w'_1, w'_2 \rangle \in \mathcal{R}(\mathcal{C})$.

With both $\langle w_0, w'_1 \rangle \in \mathcal{R}(\mathcal{B})$ and $\langle w'_1, w'_2 \rangle \in \mathcal{R}(\mathcal{C})$, we have $\langle w_0, w'_2 \rangle \in \mathcal{R}(\mathcal{B}) \circ \mathcal{R}(\mathcal{C}) = \mathcal{R}(\mathcal{B}|\mathcal{C})$. From the definition of τ_2, we know that w'_2 is in $\mathcal{E}(\sigma)$ exactly when $\sigma \in \overline{w}_2$ for all $\sigma \in T$, so $\overline{w}_2 = \phi_T(w'_2)$. We have shown that $\overline{w}_2 \in \phi^w_T(\mathcal{R}(\mathcal{B}|\mathcal{C})(w_0))$, and therefore that given the hypothesis, the model supports $\mathcal{B}|\mathcal{C} \overset{T}{\Rightarrow} \mathcal{A}|\mathcal{C}$. □

Axiom E17. This axiom follows from our brute-force semantics for names. Assume the premise:

$$\mathcal{R}(\mathcal{A}) \subseteq \mathcal{R}(\mathcal{B})$$

We want to show that

$$\mathcal{R}(\mathcal{A} \cdot N) \subseteq \mathcal{R}(\mathcal{B} \cdot N),$$

which is of course trivial thanks to requirement I of Definition E21.

Theorem E18. Since $(\mathcal{A} \wedge \mathcal{B}) \Rightarrow \mathcal{A}$, $(\mathcal{A} \wedge \mathcal{B}) \cdot N \Rightarrow \mathcal{A} \cdot N$ (by Axiom E17, with $T = \mathcal{U}$). The same is true for $\mathcal{B}$, proving:

$$(\mathcal{A} \wedge \mathcal{B}) \cdot N \Rightarrow (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \qquad \square$$

Axiom E19. Requirement III of Definition E21 exists to support this axiom. It says:

$$\mathcal{R}(\mathcal{A} \wedge \mathcal{B}) \cdot N \subseteq \mathcal{R}(\mathcal{A} \cdot N) \cup \mathcal{R}(\mathcal{B} \cdot N)$$

The right-hand side, by the semantics for $\wedge$, is equal to $\mathcal{R}((\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N))$, completing the proof.

Theorem E20. Theorem E18 and Axiom E19 together show equality. □

Axiom E23. Assume $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma') \sqsupset \mathcal{R}(\mathcal{A}) \subseteq \mathcal{E}(\sigma')$ for $\sigma' \in \{\sigma, \tau\}$. Further, assume that $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma \wedge \tau)$. Using the semantics of $\wedge$, we can write $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma) \cap \mathcal{E}(\tau)$, and hence $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma)$ and $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\tau)$. By the first assumption, we can replace $\mathcal{B}$ in both statements with $\mathcal{A}$, use the definition of $\cap$ and the semantics of $\wedge$, and conclude that $\mathcal{R}(\mathcal{A}) \subseteq \mathcal{E}(\sigma \wedge \tau)$, justifying the axiom. □

Axiom E24. Let $T = \{\sigma, \tau\}$ and $T' = \{\sigma \wedge \tau\}$. Assume first that:

$$\phi_T^w(\mathcal{R}(\mathcal{A})(w_0')) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w_0')) \;\forall\, w_0' \in W$$

Second, assume we are given w_0 and $\overline{w}_1'$ such that $\overline{w}_1' \in \phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0))$. We have the existence of a $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$ with $\overline{w}_1' = \phi_{T'}(w_1)$.

Let $\overline{w}_1 = \phi_T(w_1)$. By our first assumption, $\overline{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$, so there is a $w_1' \in \mathcal{R}(\mathcal{B})(w_0)$ with $\overline{w}_1 = \phi_T(w_1')$. We claim that $\phi_{T'}(w_1') = \overline{w}_1'$, a claim supported by leaning on the definition of ϕ_T:

$$\begin{aligned}
\sigma \wedge \tau \in \phi_{T'}(w_1') &\equiv w_1' \in \mathcal{E}(\sigma \wedge \tau) \\
&\equiv w_1' \in \mathcal{E}(\sigma) \wedge w_1' \in \mathcal{E}(\tau) \\
&\equiv \sigma \in \overline{w}_1 \wedge \tau \in \overline{w}_1 \\
&\equiv w_1 \in \mathcal{E}(\sigma) \wedge w_1 \in \mathcal{E}(\tau) \\
&\equiv w_1 \in \mathcal{E}(\sigma \wedge \tau) \\
&\equiv \sigma \wedge \tau \in \overline{w}_1'
\end{aligned}$$

Since $\overline{w}_1'$ is either $T = \{\sigma \wedge \tau\}$ or $\varnothing$, we have shown the equality, and that $\overline{w}_1' \in \phi_{T'}^w(\mathcal{R}(\mathcal{B})(w_0))$. Therefore the model supports $B \overset{\{\sigma\wedge\tau\}}{\Rightarrow} A$. □

Axiom E25. The structure of this proof parallels that of Axiom E24. Let $T = \{\sigma\}$ and $T' = \{\neg\sigma\}$. Assume first that:

$$\phi_T^w(\mathcal{R}(\mathcal{A})(w_0')) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w_0')) \;\forall\, w_0' \in W$$

Second, assume we are given w_0 and $\overline{w}_1'$ such that $\overline{w}_1' \in \phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0))$. That implies the existence of a $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$, with $\overline{w}_1' = \phi_{T'}(w_1)$. By the definition of $\phi_{T'}$ we know $w_1 \in \mathcal{E}(\neg\sigma)$ if and only if $\neg\sigma \in \overline{w}_1'$. Using the semantics of $\neg$, we can rewrite that expression as

$$w_1 \in \mathcal{E}(\sigma) \;\textbf{ iff }\; \neg\sigma \notin \overline{w}_1'$$

Define

$$\overline{w}_1 = \begin{cases} T & \text{if } \overline{w}_1' = \varnothing \\ \varnothing & \text{otherwise } (\overline{w}_1' = T') \end{cases}$$

Clearly $\sigma \in \overline{w}_1$ if and only if $\neg\sigma \notin \overline{w}_1'$. Now we can write

$$w_1 \in \mathcal{E}(\sigma) \;\textbf{ iff }\; \sigma \in \overline{w}_1$$

This expression satisfies the definition of ϕ_T, so we have $\phi_T(w_1) = \overline{w}_1$. Because $w_1 \in \mathcal{R}(A)(w_0)$, we know $\overline{w}_1 \in \phi_T^w(\mathcal{R}(A)(w_0))$.

Using the first assumption, we have $\overline{w}_1 \in \phi_T^w(\mathcal{R}(B)(w_0))$. Using arguments analogous to those above, we have the existence of a $w_1' \in \mathcal{R}(B)(w_0)$, and by the definition of ϕ_T, we can show that $\overline{w}_1'$ is in $\phi_T^w(\mathcal{R}(B)(w_0))$ as well. The model supports $\mathcal{B} \overset{\neg\sigma}{\Rightarrow} \mathcal{A}$. □

A.5 Relationships among the restricted relations

In each of the examples below, assume $T = \{s\}$.

$\overset{T}{\Rrightarrow}$ **is not stronger than** $\overset{T}{\Rightarrow}$**.** The subset relation in the projected model $\overline{M}$ of $\overset{T}{\Rightarrow}$ holds with the possible exception of the single world $\overline{w}_T = T$ that represents the equivalence class of worlds in $\mathcal{M}$ in which all statements in T hold. Clearly ϕ_T takes every member of $\cap_{\sigma\in T}\mathcal{E}(\sigma)$ to that representative. The counterexample illustrated in Figure 10 highlights this exception.

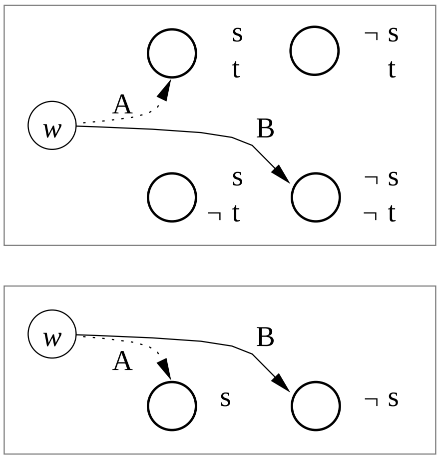
$\overset{T}{\Rightarrow}$ **is not stronger than** $\overset{T}{\Rrightarrow}$**.** Although just showed that $\overset{T}{\Rrightarrow}$ is not quite stronger than $\overset{T}{\Rightarrow}$, it certainly seems almost so. Indeed, it is very easy to construct an example that shows that the mighty relation does not follow from the basic speaks-for-regarding relation. See Figure 11.

$\overset{T}{\Rightarrow}$ **implies** $\overset{T}{\rightarrow}$**.** Assume $\mathcal{R}(A) \subseteq \phi_T^+(\mathcal{R}(B))$. We will prove by contradiction that $B \overset{T}{\rightarrow} A$. To establish a contradiction, we assume there is a statement $\sigma \in T$ and a world w_0 where $B\,\textbf{says}\,\sigma$ but not $A\,\textbf{says}\,\sigma$. That is, $\mathcal{R}(B)(w_0) \subseteq \mathcal{E}(\sigma)$ but $\mathcal{R}(A)(w_0) \not\subseteq \mathcal{E}(\sigma)$. The latter means that there is a world $w_1 \in \mathcal{R}(A)(w_0)$, but $w_1 \notin \mathcal{E}(\sigma)$.

We can push $\langle w_0, w_1\rangle$ through our original assumption to find a w_1' such that $\langle w_0, w_1'\rangle \in \mathcal{R}(B)$ and $w_1' \cong_T w_1$. Definition E11 tells us that $w_1' \notin \mathcal{E}(\sigma)$, which means $\mathcal{R}(B)(w_0) \not\subseteq \mathcal{E}(\sigma)$, which contradicts our second assumption. We may conclude that for all $w_0 \in W$ and $\sigma \in T$, $\mathcal{R}(B)(w_0) \subseteq \mathcal{E}(\sigma)$ implies $\mathcal{R}(A)(w_0) \subseteq \mathcal{E}(\sigma)$. □

$\overset{T}{\Rrightarrow}$ **implies** $\overset{T}{\rightarrow}$**.** We assume

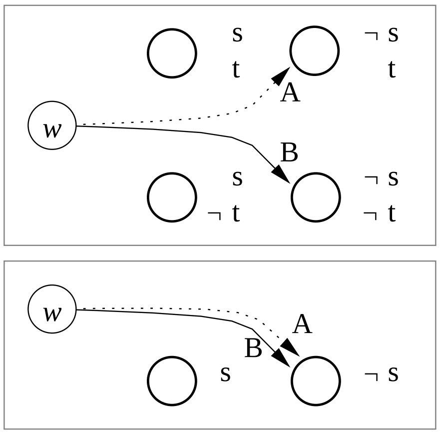
$$\mathcal{R}(A)(w_0) - \bigcap_{\tau\in T} \mathcal{E}(\tau) \subseteq \mathcal{R}(B)(w_0)$$



The set $\cap_{s \in T} \mathcal{E}(s)$ is the left pair of worlds (where s is true); the only edge belonging to $\mathcal{R}(A)$ terminates in one of those worlds. Therefore, in this model, $\mathcal{R}(A)(w) - \cap_{s \in T} \mathcal{E}(s) \subseteq \mathcal{R}(B)(w)$, and we conclude that $B \overset{T}{\Rrightarrow} A$.

The mapping ϕ_T that reduces the worlds above to equivalence classes modulo statements in T will make this model $\mathcal{M}'$. $\phi_T^R(\mathcal{R}(A))$ includes an edge to the equivalence class labeled s, but $\phi_T^R(\mathcal{R}(B)(w))$ does not. Therefore, $B \overset{T}{\not\Rightarrow} A$.

Figure 10: *A counterexample that shows* $B \overset{T}{\Rrightarrow} A$ *does not imply* $B \overset{T}{\Rightarrow} A$.



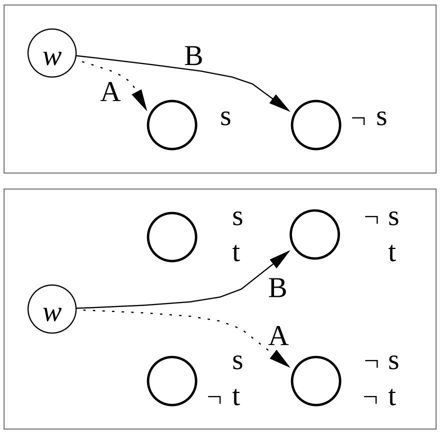
Here is a model in which from w, A considers possible a world neither in $\mathcal{R}(B)(w)$ nor $\cap_{s \in T} \mathcal{E}(s)$. So $B \overset{T}{\not\Rrightarrow} A$.

Projecting the model onto T, however, shows that $\phi_T^R(\mathcal{R}(A))$ and $\phi_T^R(R(B))$ completely agree on matters related to s; that is, $B \overset{T}{\Rightarrow} A$.

Figure 11: *A counterexample that shows* $B \overset{T}{\Rightarrow} A$ *does not imply* $B \overset{T}{\Rrightarrow} A$.

and that $\mathcal{R}(B)(w_0) \subseteq \mathcal{E}(\sigma)$. From the first assumption, any world $w_1 \in \mathcal{R}(A)(w_0)$ is either in $\mathcal{E}(\sigma)$ (let $\tau = \sigma$) or in $\mathcal{R}(B)(w_0)$. The former case trivially guarantees $w_1 \in \mathcal{E}(\sigma)$, and the latter case does so by the second assumption. We conclude that $\mathcal{R}(A)(w_0) \subseteq \mathcal{E}(\sigma)$. □

$\overset{T}{\rightarrow}$ is weaker than $\overset{T}{\Rightarrow}$ and $\overset{T}{\rightarrow}$ is weaker than $\overset{T}{\Rrightarrow}$. See Figure 12 for counterexamples that illustrate these relationships.



(a) The statement $(\mathcal{R}(B)(w) \subseteq \mathcal{E}(s)) \supset (\mathcal{R}(A)(w) \subseteq \mathcal{E}(s))$ has a false premise, making it vacuously true in this model. Hence this model satisfies $B \xrightarrow{T} A$. The model is its own projection onto T, however, and it is clear that $B \overset{T}{\not\Rightarrow} A$.

(b) This model satisfies $B \xrightarrow{T} A$ for the same reason as the model in part (a). The single edge terminating at $\mathcal{R}(A)(w)$, however, is in neither $\mathcal{R}(B)(w)$ nor $\cap_{s \in T} \mathcal{E}(s)$, so $B \overset{T}{\not\Rrightarrow} A$.

Figure 12: *Examples that show why the relation $\xrightarrow{T}$ is weaker than $\overset{T}{\Rightarrow}$ and $\overset{T}{\Rrightarrow}$.*