15-316: Software Foundations of Security and Privacy

Lecture Notes on Authorization, Logically

Matt Fredrikson

Carnegie Mellon University
Lecture 14

1 Introduction

We have discussed several mechanisms for ensuring that programs and the principals who use them do not violate security goals. Starting with safety policies, we discussed a general approach for describing such goals using security automata that monitor the execution of a system to ensure that "bad things" that violate the policy never happen. Then we moved on to information flow policies, which can be enforced by type systems or verified using composition techniques. We saw how they are useful in preventing unwanted leakage of secret data or the incursion of untrusted data into critical parts of the system.

In both cases, we implicitly worked under a sort of closed-world assumption that allows both policy author and enforcement mechanism to assume knowledge of who will use the program (and thus who the target of enforcement is), and what the entire global policy to enforce will be. A centralized enforcement mechanism—be it a reference monitor or type checker—can then uniformly apply the policy when needed.

In many settings it is unrealistic to make such assumptions. In particular when the system in question is distributed among multiple autonomous locations and used by principals who trust eachother to varying degrees, decisions about which principals are authorized to carry out actions become complicated. The enforcement mechanism must solve two problems: learn the true identity of the requesting party (i.e., authentication), and determine whether they are allowed to make the request (i.e., authorization).

Today we will begin discussing a logical formulation of the authorization problem. The type of formalism that we will cover was pioneered by Lampson, Abadi, Burrows, and Wobber in the early 1990's [5], and is an instance of what is known as authorization logic. We will learn how to express security policies for distributed systems using authorization logic, and to reason about policy decisions using formal proof.

2 A motivating example

The doors in the CyLab portion of the CIC building are controlled by a system called Grey, which allows CyLab faculty, students, and staff to unlock rooms from a mobile app or via Bluetooth sensors on their phones. This makes it easy for CyLab to provision access to rooms in a flexible manner without having to copy and distribute physical keys. For example, each faculty member has an office, which from the perspective of the access control policy that member "owns" and should be given unconditional access to. Some faculty are given additional space for lab equipment and experimental setup, and in general CyLab may want to institute a policy where students of such faculty are given access to the lab.

If we were to formalize this, we might define several predicates: $\texttt{owns}(A, R)$ denotes that A owns room R; $\texttt{studentOf}(B, A)$ denotes that B is a student of A; and $\texttt{canOpen}(A, R)$ denotes that A can open room R. Then the most basic rule of CyLab's policy might read as shown in Equation 1.

$$P_1 \equiv \forall A.\forall R.\texttt{owns}(A, R) \rightarrow \texttt{canOpen}(A, R) \tag{1}$$

Going further, we could define a second component of CyLab's policy as shown in Equation 2.

$$P_2 \equiv \forall A.\forall B.\forall R.\texttt{owns}(A, R) \rightarrow \texttt{studentOf}(B, A) \rightarrow \texttt{canOpen}(B, R) \tag{2}$$

Then if student B of professor A wants to access the lab in room R, the device controlling the lock on R should decide whether to grant B access by constructing a proof of the judgement in Equation 3.

$$P_1, P_2, \texttt{owns}(A, R), \texttt{studentOf}(B, A) \vdash \texttt{canOpen}(B, R) \tag{3}$$

The assumptions on the left of the sequent contain CyLab's policy P_1, P_2, as well as the basic facts $\texttt{owns}(A, R), \texttt{studentOf}(B, A)$. You should be able to check that this will work as intended using the rules of first-order sequent calculus.

Perhaps the most important question to ask at this point is, who provides and maintains the basic facts like $\texttt{studentOf}(B, A)$? The most obvious answer would be the system administrator who is charged with making sure that Grey functions securely and appropriately for the CyLab residents. But is it reasonable to ask the sysadmin to keep track of who each of the approximately 100 CyLab graduate students are, and which rooms they should be granted access to? Keep in mind that students routinely join CyLab, switch advisors, and leave after graduation. This solution also requires the sysadmin to keep track of undergraduate students who help with faculty research projects for a semester, summer interns, and visiting researchers, so in a typical month the sysadmin may need to deal with adding and removing dozens of entries from the policy assumptions.

Even for an organization as small as CyLab it seems that tasking the sysadmin with full responsibility of managing authorization and identity will not scale. Needless to say, if Grey were scaled up to a larger organization, such as the entire university, then the unfortunate sysadmin's job would quickly become untenable.

3 Formalizing policy intent: A `says` P

Rather than leaving the administrator to sort all this out and make sure that it is maintained up-to-date, we will extend our logical reasoning with a new construct that expresses the intent of various principals. We use the syntax shown in (4), which represents that principal A makes statement P.

$$A \texttt{ says } P \tag{4}$$

In (4), P is a predicate that denotes some fact relevant to the authorization policy. For example, we might write `admin says canModify(Bob, /etc/passwd, modify)` to denote the fact that principal `admin` states that `Bob` is allowed to modify `/etc/passwd`. Going back to our example from before, we would re-write Equation 1 using `says` as shown in Equation 5.

$$P_1 \equiv \texttt{admin says } (\forall A.\forall R.\texttt{owns}(A,R) \rightarrow \texttt{canOpen}(A,R)) \tag{5}$$

Eq. 5 denotes a statement of policy made by the `admin` principal. Other principals can make statements as well, and indeed this was one of our motivations for introducing `says` was to distribute the job of managing priveleges among users to other trusted principals. So if Professor A wishes to allow their student B access to lab facilities owned by them, they can make the appropriate statement as shown in Eq. 6.

$$A \texttt{ says studentOf}(B,A) \tag{6}$$

Now if `admin` trusts professors, who are the only principals allowed to "own" rooms in this hypothetical scenario, to make honest statements about who their students are, then the policy line from Eq. 2 can be updated as shown in Eq. 7.

$$P_2 \equiv \texttt{admin says } (\forall A.\forall B.\forall R.\texttt{owns}(A,R) \rightarrow (A \texttt{ says studentOf}(B,A)) \rightarrow \texttt{canOpen}(B,R)) \tag{7}$$

The `says` construct is surprisingly powerful in its simplicity. For example, suppose that `admin` decided to allow those with access to a room the ability to grant others access. This can be accomplished with,

$$\texttt{admin says } ((A \texttt{ says canOpen}(B,R)) \rightarrow \texttt{canOpen}(A,R) \rightarrow \texttt{canOpen}(B,R)) \tag{8}$$

This policy says the following: `admin` states that if A says B can unlock R, then if A can unlock R, then B is allowed to unlock R.

In short, `says` allows us to write formulas that express the security-relevent intentions of various principals. This allows us to consider situations where there is no global authorization policy, but there are instead pre-existing trust relationships between the various principals.

In our running example, the endpoint devices responsible for unlocking doors in CyLab would be configured to only trust statements made by `admin`. As we will see in the rest

of the lecture, we can formulate reasoning principals for deriving such statements in the form of `says` formulas from policy formulas taken as assumptions. So when principal A attempts to unlock door R, A's phone will send a message to the device controlling the lock, to the effect of "on the authority of `admin`, I (user A) am allowed to open door R". Formally,

$$\texttt{admin says canOpen}(A, R) \tag{9}$$

The endpoint device controlling the lock will then attempt to verify this claim against its policy Γ, by a process tantamount to closing out a sequent calculus proof of the judgement shown in Equation 10.

$$\Gamma \vdash \texttt{admin says canOpen}(A, R) \tag{10}$$

We'll finish this section by pointing out that it need not be the job of the endpoint to construct a proof of Eq. 10. In fact, the way that the actual Grey implementation works [1] is an instance of proof-carrying authorization [2], wherein the principal who wishes to gain access is responsible for constructing a proof that they are authorized to do so. This is an overall "win" in terms of performance, because the party can often cache proofs for accesses that they are likely to require often, and all that the endpoint needs to do is check the proof which can be done efficiently.

4 Reasoning principles

Now that we have seen how to formalize a few basic access control policies using the `says` construct, we turn to the matter of reasoning about when access to a resource should be granted under such a policy. We will develop formal reasoning principles that allow us to construct proofs of judgements like the one in Eq. 10, with the understanding that access is to be granted exactly when one can construct a proof of an authorization statement.

In the past when we have developed proof rules and conducted proofs, our goal was concerned with the truth of propositions, and eventually the validity of formulas. When reasoning about authorization, we still need to reason about the ultimate truth of propositions, but we are also concerned with what various principals intend to affirm. Note that there is a subtle difference between what is objectively true and what an individual may be willing to affirm. While we may expect any reasonable individual to affirm any true fact, it may be problematic if we also expect individuals to refuse affirmation of facts that are not true, or at the very least cannot be supported rigorously. If we were to equate principal's affirmations with true propositions, taking them to be one and the same, we might find ourselves in the uncomfortable position of accepting absurd, contradictory propositions at face value! Needless to say, we want to be a bit more cautious when reasoning about authorization decisions.

To account for this shift in our goals, we will introduce a new type of judgement into the logic, as shown in Equation 11.

$$\Gamma \vdash A \texttt{ aff } P \tag{11}$$

Aside: When to trust **says**

Access control is about authentication and authorization. Authentication addresses the problem of identifying principals, whereas authorization deals with the question of which actions principals are allowed to take. The logic that we are discussing in today's lecture is a formal system for describing and reasoning about authorization decisions, and leaves the matter of authentication unaddressed.

It may be helpful to think of `says` as abstracting away the details of authentication, and when working with a formula A `says` P to simply assume that whatever the system in question may be, it has a reliable way of determining that A did in fact make statement P. In practice, there are a number of ways to establish such a fact. For example, the system might run on an operating system that required A to login using a password, after which A stated P on a local channel within the trusted OS. Alternatively, A may be using trusted hardware that identifies them, and is connected to a physically-secure channel between two machines.

Perhaps the most common way in which this trust is established uses cryptographic digital signatures. The details of how digital signatures work is outside the scope of this class (interested readers are referred to Chapter 12 of Katz & Lindell [4]), but for the purposes of this material it suffices to say that there are algorithms for constructing messages that 1) cannot be forged by those who do not posess a secret key; and 2) can be efficiently attributed to their originator, who holds the secret key. Such schemes can be used to sign statements like A `says` P, so that if A's secret key was used to construct the signature then others can verify that A did indeed make statement P.

The judgement in (11) reads, "from assumptions Γ, it follows that principal A affirms P." The logic that we will study includes rules for proving such judgements, in addition to the normal type of judgement that concerns the truth of a formula given assumptions. Although we simply wrote $\Gamma \vdash P$ for such judgements before, we will now disambiguate them by appending the syntax true, as shown in Equation 12.

$$\Gamma \vdash P\ \mathsf{true} \tag{12}$$

Eq. 12 is read as "from assumptions Γ, the proposition P logically follows." This is distinct from (11), which is a statement about what can be concluded about what a principal will affirm from a set of assumptions. In simple terms, (12) is a statement that is independent of the beliefs, statements, or perspective of any principal.

It is also important to note that $A\ \mathtt{aff}\ P$ is not a proposition itself, but rather a judgement that can be supported by virtue of a proof. So for example, it does not necessarily make sense to write something like Equation 13, which says that if principal A affirms proposition P then principal B does as well.

$$A\ \mathtt{aff}\ P \rightarrow B\ \mathtt{aff}\ P \tag{13}$$

However, $A\ \mathtt{says}\ P$ is a proposition–a statement that is either true or false–that we would very much like to reason about, because when making authorization decisions it is essential to know what statements principals are willing to make. Thinking back to our running example from (10), we would ultimately like to reason about whether `admin` is willing to state that A can open R.

4.1 Relating `says` and `aff`

So how do we relate the judgements $(A\ \mathtt{says}\ P)\ \mathsf{true}$ and $A\ \mathtt{aff}\ P$? The first rule SaysR that we introduce for doing so lets us reason that if A affirms P under assumptions Γ, then A is willing to state A under those assumptions.

$$\text{(SaysR)}\quad \frac{\Gamma \vdash A\ \mathtt{aff}\ P}{\Gamma \vdash (A\ \mathtt{says}\ P)\ \mathsf{true}}$$

The rule SaysR tells us how to reason when $A\ \mathtt{says}\ P$ is on the right side of a sequent, and our goal is to prove the truth of such a statement. The reasoning principle embodied in this rule requires that in order to prove $A\ \mathtt{says}\ P$ under Γ, we must be able to prove $A\ \mathtt{aff}\ P$ under Γ. In other words, principals are willing to state anything that they affirm.

How do we make use of `says` formulas, i.e., what do we do when $A\ \mathtt{says}\ P$ appears on the left side of a sequent? One possibility is simply to conclude that if A states P, then $P\ \mathsf{true}$. But after some thought, this rule is not satisfactory, because it allows principals to fabricate truth from statements, which we determined earlier would be problematic. What would happen if our policy context had principals that stated contradictory things? For example, this could happen if one were to assume (14) in addition to the Grey policies P_1, P_2 from Eqs. 5,7.

$$\mathtt{studentOf}(B, A), \mathtt{owns}(A, R), \mathtt{canOpen}(B, R) \rightarrow \mathit{false} \tag{14}$$

We certainly do not want to admit proofs of *false* true, and requiring that principals only say true things seems difficult to enforce, to say the least. Instead, if we have the assumption $(A\ \mathtt{says}\ P)\ \mathsf{true}$, then perhaps we can more reasonably use it to conclude that A affirms some other proposition Q, $A\ \mathtt{aff}\ Q$. But when should we be able to conclude that $A\ \mathtt{aff}\ Q$ from assumption $(A\ \mathtt{says}\ P)\ \mathsf{true}$? We can reason that if A is willing to affirm Q assuming that A is willing to state P, then requiring a proof of $P\ \mathsf{true}$ to justify A's statement is the logical requirement. This is captured in the rule SaysL below.

$$(\text{SaysL})\quad \frac{\Gamma, P\ \mathsf{true} \vdash A\ \mathtt{aff}\ Q}{\Gamma, (A\ \mathtt{says}\ P)\ \mathsf{true} \vdash A\ \mathtt{aff}\ Q}$$

Now we have left and right rules for $\mathtt{says}$ formulas. Looking at them together, we can see that their premises work together in a useful way that lets us reason about the affirmations of a principal. Namely, if $\Gamma \vdash A\ \mathtt{aff}\ P$ and $\Gamma, A\ \mathsf{true} \vdash A\ \mathtt{aff}\ Q$, then the rules let us reason that $\Gamma \vdash A\ \mathtt{aff}\ Q$. This should remind you of the cut rule from propositional sequent calculus earlier in the semester, as it captures similar reasoning for affirmations.

Finally, we must address the question of when it is appropriate to conclude that A affirms a statement P in the first place. Note that SaysL doesn't quite solve this problem, as the obligation that it introduces itself contains an affirmation judgement. The rule Aff below says that any principal is willing to affirm true propositions.

$$(\text{Aff})\quad \frac{\Gamma \vdash P\ \mathsf{true}}{\Gamma \vdash A\ \mathtt{aff}\ P}$$

Together with SaysL and SaysR, Aff gives us everything necessary to reason about authorization formulas.

4.2 Revisiting connectives

We could add the term $A\ \mathtt{says}\ P$ to first-order predicate calculus, and use the rules from the previous section along with all the others that we learned earlier in the semester. But consider the judgement in Equation 15, which says that if A says P, then either P must be true or A is willing to state *false*.

$$((A\ \mathtt{says}\ P) \to (\neg P \to (A\ \mathtt{says}\ \mathit{false})))\ \mathsf{true} \tag{15}$$

This seems like a bad theorem to have in our logic, because it places a strong burden on principals: either they must always say true things, or if they don't then we must conclude that they are willing to say anything—even *false*! But is this a theorem? Let's see if we can derive a proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P\ \mathsf{true} \vdash P\ \mathsf{true}, A\ \mathtt{aff}\ \mathit{false}}\,\text{id}}{P\ \mathsf{true}, \neg P\ \mathsf{true} \vdash A\ \mathtt{aff}\ \mathit{false}}\,\neg\text{L}}{(A\ \mathtt{says}\ P)\ \mathsf{true}, \neg P\ \mathsf{true} \vdash A\ \mathtt{aff}\ \mathit{false}}\,\text{SaysL}}{(A\ \mathtt{says}\ P)\ \mathsf{true}, \neg P\ \mathsf{true} \vdash (A\ \mathtt{says}\ \mathit{false})\ \mathsf{true}}\,\text{SaysR}}{\vdash ((A\ \mathtt{says}\ P) \to (\neg P \to (A\ \mathtt{says}\ \mathit{false})))\ \mathsf{true}}\,{\to}\text{R},{\to}\text{R}$$

$$(\bot\text{L})\ \frac{}{\Gamma, \bot\ \mathsf{true} \vdash P\ \mathsf{true}} \qquad (\text{id})\ \frac{}{\Gamma, P\ \mathsf{true} \vdash P\ \mathsf{true}}$$

$$(\to\text{L})\ \frac{\Gamma \vdash P\ \mathsf{true} \quad \Gamma, Q\ \mathsf{true} \vdash R\ \mathsf{true}}{\Gamma, (P \to Q)\ \mathsf{true} \vdash R\ \mathsf{true}} \qquad (\to\text{R})\ \frac{\Gamma, P\ \mathsf{true} \vdash Q\ \mathsf{true}}{\Gamma \vdash (P \to Q)\ \mathsf{true}}$$

$$(\forall\text{L})\ \frac{\Gamma, F(e)\ \mathsf{true} \vdash P\ \mathsf{true}}{\Gamma, \forall x.F(x)\ \mathsf{true} \vdash P\ \mathsf{true}} \qquad (\forall\text{R})\ \frac{\Gamma \vdash F(y)\ \mathsf{true}}{\Gamma \vdash \forall x.F(x)\ \mathsf{true}}(y\text{ new})$$

$$(\text{SaysL})\ \frac{\Gamma, P\ \mathsf{true} \vdash A\ \mathtt{aff}\ Q}{\Gamma, (A\ \mathtt{says}\ P)\ \mathsf{true} \vdash A\ \mathtt{aff}\ Q} \qquad (\text{SaysR})\ \frac{\Gamma \vdash A\ \mathtt{aff}\ P}{\Gamma \vdash (A\ \mathtt{says}\ P)\ \mathsf{true}}$$

$$(\text{Aff})\ \frac{\Gamma \vdash P\ \mathsf{true}}{\Gamma \vdash A\ \mathtt{aff}\ P}$$

Figure 1: Proof rules for authorization logic of Garg and Pfenning [3].

Indeed we are able to derive a proof, so it seems that if we allow our rules from before, or at the very least →R, ¬L, then we would have to accept this consequence.

An alternative was proposed by Garg and Pfenning in 2006 [3]. They pointed out that the above is actually a consequence of admitting the law of excluded middle (LEM) into the logic, which says that $P \vee \neg P$. In particular, LEM is a consequence of our right negation and implication rules, and the fact that sequents can contain multiple formulas on the right as a disjunction. They proposed a constructive authorization logic, in which LEM is not a theorem. The syntax of their logic is shown in (16), and the proof rules are given in Figure 1. Note that each of the sequents can have multiple assumptions, reflected in Γ as before, but is restricted to having exactly one consequent.

$$P, Q ::= p(e_1, \ldots, e_n) \mid \mathit{true} \mid \mathit{false} \mid P \to Q \mid \forall x.P \mid A\ \mathtt{says}\ P \tag{16}$$

4.3 Back to the example

Now that we have proof rules to reason about authorization decisions, let us return to our example about CyLab's Grey system. To keep our judgements less cluttered, we will now drop the true notation from standard judgements of truth whenever no confusion will arise between them and affirmation judgements.

Recall that our system-wide policy was comprised of two formulas of authorization logic. The first P_1 reflects that the administrator states that anyone who owns a room is allowed to unlock it.

$$P_1 \equiv \mathtt{admin\ says}\ (\forall A.\forall R.\mathtt{owns}(A, R) \to \mathtt{canOpen}(A, R))$$

The second says that students of faculty who own a room are also allowed to unlock that

room.

$$P_2 \equiv \texttt{admin says } (\forall A.\forall B.\forall R.\texttt{owns}(A,R) \\ \rightarrow (A \texttt{ says studentOf}(B,A)) \rightarrow \texttt{canOpen}(B,R))$$

We'll label a few more subformulas just to keep the proof uncluttered.

$$P_3 \equiv \forall A.\forall B.\forall R.\texttt{owns}(A,R) \rightarrow (A \texttt{ says studentOf}(B,A)) \rightarrow \texttt{canOpen}(B,R)$$
$$P_4(A,B,R) \equiv \texttt{owns}(A,R) \rightarrow (A \texttt{ says studentOf}(B,A)) \rightarrow \texttt{canOpen}(B,R)$$
$$P_5(A,B,R) \equiv (A \texttt{ says studentOf}(B,A)) \rightarrow \texttt{canOpen}(B,R)$$

So putting this all together, the context from which Grey will attempt to conduct authorization proofs is shown in Eq. 17.

$$\Gamma \equiv P_1, P_2 \tag{17}$$

We also denote the following formulas Q_1, Q_2 which are needed in this example.

$$Q_1 \equiv \texttt{owns}(\texttt{mfredrik}, \texttt{cic2126})$$
$$Q_2 \equiv \texttt{mfredrik says studentOf}(\texttt{alice}, \texttt{mfredrik})$$

Suppose that Alice needs to access `cic2126` to pick up the final exams before class. Alice has Grey installed on her phone, so she should be able to do this as long as her phone is able to construct a proof of `admin says canOpen(alice, cic2126)`. Let's begin constructing this proof.

The first step is rather obvious, as we want to prove a `says` formula on the right of the sequent, so SaysR is our only option.

$$\text{SaysR}\frac{P_1, P_2 \vdash \texttt{admin aff canOpen}(\texttt{alice}, \texttt{cic2126})}{P_1, P_2 \vdash \texttt{admin says canOpen}(\texttt{alice}, \texttt{cic2126})}$$

But after this we need to think a bit before deciding which rule to apply next. Because the principal attempting to open the door is `alice`, who is using the fact that `mfredrik` owns it to do so, it seems that P_2 is the relevant policy to operate on. To access it, we need to first apply SaysL to eliminate the `admin says` around it, followed by ∀L using $A \mapsto \texttt{mfredrik}, B \mapsto \texttt{alice}, R \mapsto \texttt{cic2126}$. Continuing from where we left off before, this gives us the following deduction.

$$\text{SaysL}\frac{\forall\text{L}\dfrac{P_1, P_4(\texttt{mfredrik}, \texttt{alice}, \texttt{cic2126}) \vdash \texttt{admin aff canOpen}(\texttt{alice}, \texttt{cic2126})}{P_1, P_3 \vdash \texttt{admin aff canOpen}(\texttt{alice}, \texttt{cic2126})}}{P_1, P_2 \vdash \texttt{admin aff canOpen}(\texttt{alice}, \texttt{cic2126})}$$

Now $P_4(\texttt{mfredrik}, \texttt{alice}, \texttt{cic2126})$ corresponds to the formula in (18).

$$\texttt{owns}(\texttt{mfredrik}, \texttt{cic2126}) \\ \rightarrow (\texttt{mfredrik says studentOf}(\texttt{alice}, \texttt{mfredrik})) \rightarrow \texttt{canOpen}(\texttt{alice}, \texttt{cic2126}) \tag{18}$$

We want to get at the final conclusion of this implication chain. We apply →L twice on P_4. After the first application, we have the following, where the elided formula is `admin aff canOpen(alice, cic2126)`.

$$\rightarrow\text{L}\frac{P_1 \vdash \texttt{owns}(\texttt{mfredrik}, \texttt{cic2126}) \quad P_1, P_5(\texttt{mfredrik}, \texttt{alice}, \texttt{cic2126}) \vdash \ldots}{P_1, P_4(\texttt{mfredrik}, \texttt{alice}, \texttt{cic2126}) \vdash \texttt{admin aff canOpen}(\texttt{alice}, \texttt{cic2126})}$$

The second →L is applied to P_5, where the elided formula is the same as above.

$$\rightarrow\text{L}\frac{P_1 \vdash \texttt{mfredrik says studentOf}(\texttt{alice}, \texttt{mfredrik}) \quad P_1, \texttt{canOpen}(\texttt{alice}, \texttt{cic2126}) \vdash \ldots}{P_1, P_5(\texttt{mfredrik}, \texttt{alice}, \texttt{cic2126}) \vdash \texttt{admin aff canOpen}(\texttt{alice}, \texttt{cic2126})}$$

Before continuing on to close out this proof, let's first take a moment to deal with the unfinished branches that we have accumulated. The first requires that we prove `owns(mfredrik, cic2126)` from P_1, which does not follow. However, this is a fact that we can reasonably expect the system checking authorization to recognize, and have populated in its memory at the time the policy is applied. Thus, we could have included this fact in our assumptions Γ from the beginning, and closed this branch out with identity.

The second requires a proof of `mfredrik says studentOf(alice, mfredrik)`. As with the previous fact, we might assume that Grey is maintained to keep track of who each faculty claims as their student at any given time. Alternatively, `mfredrik` could provide `alice` with a cryptographically-signed statement during enrollment, so that `alice` is able to provide evidence to support the claim `mfredrik says studentOf(alice, mfredrik)` when needed. The Grey system could interactively prompt `alice`'s device for this evidence at this stage of the proof, and discharge the obligation is `alice` is able to produce an appropriately signed certificate.

Continuing on, we can close by applying Aff, which says that principals will affirm true things. We know that `canOpen(alice, cic2126)` is true because it is an assumption in the present context, so we apply id.

$$\text{Aff}\frac{\text{id}\dfrac{*}{P_1, \texttt{canOpen}(\texttt{alice}, \texttt{cic2126}) \vdash \texttt{canOpen}(\texttt{alice}, \texttt{cic2126})}}{P_1, \texttt{canOpen}(\texttt{alice}, \texttt{cic2126}) \vdash \texttt{admin aff canOpen}(\texttt{alice}, \texttt{cic2126})}$$

This finishes the proof of (`admin says canOpen(alice, cic2126)`) `true`. To summarize the general "flow" of the proof, we began by reducing our proof of what `admin` says to a proof about what `admin` will affirm. We then used the policy that `admin` has previously stated, in particular P_2 wherein `admin` said that if a professor states that someone is their student, and the professor owns a room, then the student is authorized to unlock that room. We used a combination of propositional rules from before and SaysL to reduce this policy statement into a final confirmation that `admin` affirms `canOpen(alice, cic2126)`.

5 Useful theorems

Now that we've seen how to apply the authorization logic of Figure 1 to a realistic problem, let's take a look at some general theorems of the logic itself. The first one that

we will consider is sometimes called "Unit", and says that principals are willing to say all true things.

Theorem 1 (Unit). The formula

$$P \to (A \texttt{ says } P)$$

is provable from the rules in Figure 1.

Proof. Consider the following deduction.

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{P \vdash P}\,\text{id}}{P \vdash A \texttt{ aff } P}\,\text{Aff}}{P \vdash A \texttt{ says } P}\,\text{SaysR}}{\vdash P \to (A \texttt{ says } P)}\,\to\text{R}$$

□

The next useful theorem tells us that says is closed under consequence. That is, if A says that one formula P implies another formula Q, then it follows that if A says P, A must also be willing to say Q.

Theorem 2 (Closure under consequence). The formula

$$(A \texttt{ says } (P \to Q)) \to (A \texttt{ says } P) \to (A \texttt{ says } Q)$$

is provable from the rules in Figure 1.

Proof. Consider the following deduction.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{*}{P \vdash P, Q}\,\text{id} \qquad \dfrac{*}{P, Q \vdash Q}\,\text{id}}{P \to Q, P \vdash Q}\,\to\text{L}}{P \to Q, P \vdash A \texttt{ aff } Q}\,\text{Aff}}{A \texttt{ says } (P \to Q), A \texttt{ says } P \vdash A \texttt{ aff } Q}\,\text{SaysL,SaysL}}{A \texttt{ says } (P \to Q), A \texttt{ says } P \vdash A \texttt{ says } Q}\,\text{SaysR}}{A \texttt{ says } (P \to Q) \vdash (A \texttt{ says } P) \to (A \texttt{ says } Q)}\,\to\text{R}}{\vdash (A \texttt{ says } (P \to Q)) \to (A \texttt{ says } P) \to (A \texttt{ says } Q)}\,\to\text{R}$$

□

The last theorem that we will consider tells us that says is idempotent. That is, if a principal says that they will say something (i.e., A says (A says P)), then we conclude that they are willing to just say that something to begin with.

Theorem 3 (says idempotence). The formula

$$(A \texttt{ says } (A \texttt{ says } P)) \to (A \texttt{ says } P)$$

is provable from the rules in Figure 1.

Proof. This proof is left as an exercise. □

References

[1] L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar. Device-enabled authorization in the Grey System. In Information Security, 2005.

[2] A. Chaudhuri and D. Garg. PCAL: language support for proof-carrying authorization systems. In ESORICS, pages 184–199. Springer, 2009.

[3] D. Garg and F. Pfenning. Non-interference in constructive authorization logic. In 19th IEEE Computer Security Foundations Workshop (CSFW'06), 2006.

[4] J. Katz and Y. Lindell. Introduction to Modern Cryptography, Second Edition. Chapman & Hall/CRC, 2nd edition, 2014.

[5] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. ACM Transactions on Computer Systems, 10(4), Nov. 1992.