15-316: Software Foundations of Security and Privacy

Lecture Notes on Propositional Logic and Proofs*

Matt Fredrikson

Carnegie Mellon University
Lecture 2

1 Introduction

Our foremost goal in this course is to prove that software systems obey security and privacy policies. We will cover numerous different types of policies, but in general we can think of a policy as a statement about what a program is allowed (or in some cases, not allowed) to do. If we want to actually prove things about policies, then we need to write them down precisely and in a way that allows us to use mathematical reasoning. Propositional logic is a formal system that allows us to do this for a certain types of simple policies.

For our purposes, propositional logic is a language for expressing statements (i.e., formulas) in terms of things that are either true or false (i.e., propositions) and a set of rules, called *semantics*, for determining the truth value of a formula from the truth values of its propositions. In future lectures, we will build on propositional logic to support richer and more interesting types of policies.

In this lecture, we will study propositional logic and a deductive system called the *sequent calculus* for proving the validity of propositional logic formulas. We will use the sequent calculus throughout the semester as we discuss more complex and interesting ways of proving that code is safe and secure. We will also study important properties of the propositional sequent calculus, namely its *soundness* and *completeness*. A deductive system is sound if it can only be used to prove true things. Likewise, a deductive system is complete if any true statement can be proved using the system. Throughout the semester, we will get into the habit of asking these sorts of questions about the systems we study, as they have important practical ramifications for our ultimate goal of making software secure.

*Based heavily on lecture notes contributed by André Plazter

2 A stroll down memory lane: contracts

Thinking back to 15-122 Principles of Imperative Computation, recall that contracts serve a valuable role in understanding programs. In 15-122, you made good use of contracts to specify the assumptions that functions are allowed to make of their inputs, as well as their intended behavior in terms of output values. Moreover, you learned how to reason about the behavior of imperative code by writing informal proofs about contracts, and were able to rely on dynamic checks to ensure that they weren't violated when running your code. In short, you learned that contracts are useful for making sure that your code runs correctly.

You might be surprised to learn that contracts are tremendously useful for establishing security as well. The Ironclad Apps [HHL+14] and Ironfleet [HHK+15] projects out of Microsoft Research leveraged contracts and automated verification tools to build networked applications and distributed systems with provable information flow, memory safety, data privacy, and other security properties. However, we'll refresh ourselves on contracts using a much smaller example that doesn't have much to do with security. This should be more familiar, and will serve to illustrate the main ideas that motivate our study of propositional logic.

Consider the following C0 code, which multiplies two numbers using only addition, multiplication by 2, and division by 2.

```
1  int BinaryMult(int a, int b)
2  //@requires b >= 0;
3  //@ensures  \result = a*b
4  {
5    int x=a;
6    int y=b;
7    int z=0;
8    while (y > 0)
9    //@loop_invariant 0 <= y && z + x * y = a * b;
10   {
11     if (y%2 == 1) {
12       z = z + x;
13     }
14     x = 2*x;
15     y = y/2;
16   }
17   return z;
18 }
```

This algorithm uses contracts, which is a good thing because it may not be totally obvious that this procedure does what is claimed. Are they all correct? Are they easy to follow? Is it enough to assume that `b >= 0` holds at the beginning of the function to ensure the postcondition? Does the postcondition follow easily from the loop invariant?

This is all quite exciting, but we don't yet have the tools necessary to answer these questions. The purpose of today's lecture is not actually to get us back into specifying or checking contracts of programs. Instead we'll focus on the conditions in the contract and try to understand exactly what they mean, and how we can reason about

them. Our reading in the 15-122 course was that the C0 contracts `@requires`, `@ensures`, `@loop_invariant` and `@assert` just expect ordinary C0 expressions of type `bool` that are being evaluated and need to come back with value `true` to successfully pass. But what exactly does the expression `\result` mean in the `@ensures` postcondition? What if the C0 expression in a contract calls a function that has the side effect of changing a data structure? What exactly is the meaning of the `&&` operator itself? Does it perform short-circuit evaluation? What if an expression crashes or doesn't terminate during contract evaluation?

These are quite a number of subtle questions for something that we thought we had already mastered as well as the contracts from Principles of Imperative Computation.

3 Propositional Logic

Maybe we should first take a step back and give the expressions within a contract a more careful look to see how they can best be understood. We'll start with propositional logic, which will allow us to understand the basic logical connectives used in contracts. Once we have a better understanding of these fundamentals, we will return to the interesting questions raised in the example from before.

3.1 Syntax

The objects that we will study in propositional logic are called formulas, and are composed of the following elements.

Atomic Propositions. The basic building blocks are propositions, which you can view as variables that take Boolean (i.e., true/false) values. Some might find it helpful to think of propositions as statements such as "The memory at address `0x00105f0` holds the value 10", which are either true or false. However, we won't bother interpreting the meaning of such associations for now, and we'll just denote atomic propositions with lowercase letters and treat them as abstract statements that could be either true or false.

Connectives. Propositional formulas may contain the connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, which are used to construct formulas from propositions and other formulas.

We define the ways in which propositions, connectives, and formulas can be syntactically combined by writing a grammar as shown in Figure 1.

Definition 1 (Syntax of propositional logic)**.** The formulas F, G of propositional logic are defined by the following grammar (where p is an atomic proposition):

$$F ::= \bot \mid \top \mid p \mid \neg F \mid F \wedge G \mid F \vee G \mid F \rightarrow G \mid F \leftrightarrow G$$

In Definition 1, $\bot$ and $\top$ stand for the constants false and true, respectively. The way to read such a grammar is as follows:

- The Boolean constants $\bot$ and $\top$ are formulas.
- An atomic proposition (usually denoted p, q, r) is a formula.
- If F is a formula, then its negation $\neg F$ is also a formula.
- If F and G are formulas, then the conjunction $F \wedge G$, the disjunction $F \vee G$, the implication $F \to G$, and the bisubjunction $F \leftrightarrow G$ are also formulas.

Parentheses are also allowed as needed to make precedence explicit. For example, this is a propositional formula:

$$(p \to q) \leftrightarrow (\neg p \vee q) \tag{1}$$

We'll use the following precedence on operators, from highest to lowest: $\neg, \wedge, \vee, \to, \leftrightarrow$. We will also assume that $\to$ and $\leftrightarrow$ associate to the right, so the following:

$$t \wedge p \to q \to r \to s \tag{2}$$

is equivalent to:

$$(t \wedge p) \to (q \to (r \to s)) \tag{3}$$

Parentheses are cumbersome, so we'll avoid using them whenever possible.

4 Semantics

Writing down logical formulas that fit to the syntax of propositional logic is one thing, but not particularly useful unless we also know whether the formulas true. We cannot generally know whether the atomic propositions in a propositional logical formula are true or false, because they are just called p, q, r, which does not tell us much about their intention. But we can define some structure to help us out. Let's fix a function I, called the *interpretation*, that tells us the truth-value for each atomic proposition. So $I(p) = \top$ iff atomic proposition p is interpreted as true in interpretation I. For example, we could fix the following interpretation for formula (1):

$$I = \{q\} \tag{4}$$

By this notation, we mean the interpretation that satisfies $I(q) = \top$ and interprets all other atomic propositions (i.e., just p in this case) as $\bot$.

Having fixed an interpretation I for the atomic proposition, we can now easily evaluate all propositional formulas to see whether they are true or false in that interpretation I of atomic propositions, because the logical operators $\wedge, \vee, \neg, \to, \leftrightarrow$ always have exactly the same meaning.

Definition 2 (Semantics of propositional logic)**.** The propositional formula F is true in interpretation I, written $I \models F$, as inductively defined by distinguishing the shape of formula F:

1. $I \not\models \bot$, i.e., $\bot$ is true in no interpretations

2. $I \models \top$, i.e., $\top$ is true in all interpretations
3. $I \models p$ iff $I(p) = \top$ for atomic propositions p
4. $I \models F \wedge G$ iff $I \models F$ and $I \models G$.
5. $I \models F \vee G$ iff $I \models F$ or $I \models G$.
6. $I \models \neg F$ iff $I \not\models F$, i.e. it is not the case that $I \models F$.
7. $I \models F \rightarrow G$ iff $I \not\models F$ or $I \models G$.
8. $I \models F \leftrightarrow G$ iff both are true or both false, i.e., it is either the case that both $I \models F$ and $I \models G$ or it is the case that $I \not\models F$ and $I \not\models G$.

With this definition, it is easy to establish that formula (1) is true in interpretation (4):

$$I \models (p \rightarrow q) \leftrightarrow (\neg p \vee q)$$

For example, the evaluation of the right-hand side formula after the implication $\rightarrow$ proceeds as follows:

$$I \models p \rightarrow q \text{ because } I \not\models p \text{ because } I(p) = \bot$$

Now we can ask more interesting questions about formula (1) and others, like: is formula (1) only true in this particular interpretation, or what happens with other interpretations of p, q that assign different Boolean values?

The most exciting formulas are those that are true no matter what the interpretation of the atomic propositions is. Such a formula is called *valid* because it expresses a true property regardless of the specific interpretation of the atomic propositions. We will also talk about *satisfiable* formulas, which are those for which there is at least one interpretation that makes the formula true.

Definition 3 (Validity & Satisfiability)**.** A formula F is called *valid* iff it is true in all interpretations, i.e. $I \models F$ for all interpretations I. Because any interpretation makes valid formulas true, we also write $\vDash F$ iff formula F is valid. A formula F is called *satisfiable* iff there is an interpretation I in which it is true, i.e. $I \models F$. Otherwise it is called *unsatisfiable*.

You may wonder what exactly the relationship between satisfiability and validity is. Most obviously, if F is valid then it is also satisfiable; likewise, if F is unsatisfiable then it is certainly not valid. But there is more to this connection. Suppose that F is valid, so for any interpretation $I \models F$. By the semantics of negation, $I \not\models \neg F$, so it must be the case that $\neg F$ is unsatisfiable. Conversely, suppose that $\neg F$ is unsatisfiable. Then for any interpretation I, $I \not\models \neg F$, and by the semantics of negation $I \models F$. So F is valid whenever $\neg F$ is unsatisfiable.

In this sense, satisfiability and validity are duals of eachother, and a statement about the validity (resp. satisfiability) of a formula is also one about the satisfiability (resp.

validity) of its negation. To make this explicit, we can summarize by saying F *is valid if and only if* $\neg F$ *is unsatisfiable*.

It is not difficult to imagine how we might decide whether a propositional formula is valid; we simply try *all* interpretations of the atomic propositions, using the semantics to decide whether the formula is true. Let's tabulate our results for Eq. 1 by writing down each combination of truth-values for all atomic propositions and evaluating all subformulas of (1) according to their semantics.

p	q	$p \to q$	$\neg p$	$\neg p \vee q$	(1)
$\top$	$\top$	$\top$	$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$
$\bot$	$\bot$	$\top$	$\top$	$\top$	$\top$

Indeed, the truth-value of the formula (1) is $\top$ in all interpretations, thus, (1) is valid:

$$\vDash (p \to q) \leftrightarrow (\neg p \vee q)$$

The only downside is all this busywork to evaluate all interpretations, which is exponential in the number of atomic propositions and incredibly boring on top of that.

5 Proofs for Propositional Logic

Brute-force evaluating a formula in all possible interpretations is certainly one way of establishing that a propositional logical formula is valid, but it always requires exponential effort and is quite uninsightful, because it does not even attempt to provide a comprehensible reason for the validity of the formula. The only way to check that a truth table is constructed correctly for a formula is to check that it enumerates all cases of interpretations and all its computations of truth-values are according to the semantics and that, indeed, $\top$ is the outcome in all cases. Possible but incredibly dull. Besides, this finite enumeration principle cannot work for the more interesting and expressive logics that we will use to describe security properties in subsequent lectures.

The semantics considered one operator at a time. Let's try to make the same thing happen for proofs as well. What about a proof of a conjunction $F \wedge G$? A proof of a conjunction $F \wedge G$ should consist of a proof of the left conjunct F together with a proof of the right conjunct G, because both proofs together prove the conjunction $F \wedge G$. So stapling a proof of F together with a proof of G will give us a proof of $F \wedge G$. That was easy enough.

But what does a proof of an implication $F \to G$ consist of? It certainly isn't a proof of F together with a proof of G anymore. A proof of G would constitute a proof of $F \to G$, but such a proof is missing out on an important power. It would have been allowed to assume F, because the formula $F \to G$ only says that F implies G, so that G is true in case F is. If F isn't true, then the implication $F \to G$ doesn't say anything about whether G is true or not. Consequently, an unconditional proof of G certainly does establish $F \to G$, but is a bit much to ask for. The proof of $F \to G$ should, instead,

consist of a proof of G that is allowed to assume F. This requires the capability to manage assumptions in a proof, which, retrospectively, should not actually come as a surprise.

For managing assumptions in a structured way, we will follow in the footsteps of Gerhard Gentzen [Gen35], who introduced sequent calculus for the study of logic. But it turns out that sequent calculi are also immensely useful not just for understanding logical reasoning, but also for organizing and conducting proofs without risking to lose track of assumptions.

5.1 Simple Sequents

The first kind of *sequent* that we will consider is of the form:

$$\Gamma \vdash F$$

The available assumptions are given as a list of formulas Γ as *antecedent* and with the formula we want to prove from them as F, the *succedent*. The symbol $\vdash$ is called *sequent turnstile* and separates the available assumptions from what we try to prove from them.

There are some sequents where we are obviously done with a proof. For example when literally the same formula F is in the antecedent and the succedent, because F easily follows when assuming F. So the sequent $\Gamma, F \vdash F$ has a trivial proof. We will later capture this thought with a proof rule id, but first consider proofs for the operators we've already seen.

Coming back to conjunctions, proving a conjunction $F \wedge G$ requires proving F and proving G. This fact does not change when working from a list of assumptions Γ.

$$(\wedge\mathrm{R})\ \frac{\Gamma \vdash F \quad \Gamma \vdash G}{\Gamma \vdash F \wedge G}$$

This proof rule $\wedge$R expresses that all it takes to prove the *conclusion* $\Gamma \vdash F \wedge G$ below the rule bar is to prove all the *premises* $\Gamma \vdash F$ and $\Gamma \vdash G$ above the rule bar. In the proof of the left premise $\Gamma \vdash F$, the same assumptions Γ will still be available that were available in the conclusion $\Gamma \vdash F \wedge G$. And likewise for the right premise.

Proving an implication $F \to G$, with which we had difficulties before, now simply allows us to add the assumption F to the antecedent with the list of all available assumptions and continue a proof of G from this augmented list of assumptions:

$$(\to\mathrm{R})\ \frac{\Gamma, F \vdash G}{\Gamma \vdash F \to G}$$

Reading the rule $\to$R from bottom to top means that a proof of an implication $F \to G$ from a list of assumptions Γ requires us to prove G from the assumptions Γ together with F.

Proving a disjunction $F \vee G$ is more subtle. How do we prove a disjunction? We could prove a disjunction $F \vee G$ by proving the left disjunct F:

$$(\vee\mathrm{R}_1)\ \frac{\Gamma \vdash F}{\Gamma \vdash F \vee G}$$

That works. But then what if the disjunction $F \vee G$ is true because the right disjunct G is true? Well, we could adopt yet another proof rule for disjunction that shows the right disjunct instead:

$$(\vee\mathrm{R}_2)\ \frac{\Gamma \vdash G}{\Gamma \vdash F \vee G}$$

This would give us a pair of proof rules $\vee\mathrm{R}_1$ and $\vee\mathrm{R}_2$ to prove disjunctions. But we will have to choose at the time of proving the disjunction $F \vee G$ whether we prove it by proving its left disjunct F with rule $\vee\mathrm{R}_1$ or whether we prove it by proving its right disjunct G with rule $\vee\mathrm{R}_2$. That requires a lot of attention when proving disjunctions. Worse yet: will we always be able to tell which disjunct we will be able to prove?

In many cases, we will be able to predict which disjunct will suffice for proof if we think ahead very carefully. But that is already not particularly helpful and convenient. Worse yet, there are cases where, for principle reasons, we will be unable to predict which disjunct of a disjunction we will prove! Suppose we are trying to prove the formula $p \vee \neg p$, which is certainly valid, because it will evaluate to $\top$ whether or not the atomic proposition p is interpreted to be $\top$. But when trying to prove the law of excluded middle $p \vee \neg p$, neither rule $\vee\mathrm{R}_1$ nor rule $\vee\mathrm{R}_2$ will succeed because the whole point of the law of excluded middle is that it will evaluate to $\top$ whether p is $\top$ or $\bot$ (so $\neg p$ is $\top$), but we cannot generally say ahead of time which side will be $\top$.

Instead, what we are going to do is to keep our options open. We will record in the sequent the fact that formulas F as well as G were both available as formulas for us to prove when proving the disjunction $F \vee G$ by keeping both as a list on the right-hand side of the sequent turnstile $\vdash$. Of course, we might have already gathered other options that we could prove, so the disjunction proof rule is:

$$(\vee\mathrm{R})\ \frac{\Gamma \vdash F, G, \Delta}{\Gamma \vdash F \vee G, \Delta}$$

Proving a disjunction $F \vee G$ from a list of assumptions Γ with a list of alternatives Δ works by splitting the disjunction into its two options F and G and continuing with a proof of the alternatives F, G, Δ from the assumptions Γ.

5.2 Sequent Calculus

To manifest this, let's properly define what a sequent $\Gamma \vdash \Delta$ is and what it means.

Definition 4 (Sequent)**.** A *sequent* $\Gamma \vdash \Delta$ organizes the reasoning into a list Γ of formulas available as assumptions, called *antecedent*, and a list Δ called *succedent*. The semantics of sequent $\Gamma \vdash \Delta$ is the same as that of the formula

$$\left(\bigwedge_{F\in\Gamma} F\right) \rightarrow \left(\bigvee_{G\in\Delta} G\right)$$

In particular, proving a sequent $\Gamma \vdash \Delta$ requires proving that the disjunction of all succedent formulas Δ is implied by the conjunction of all antecedent formulas Γ. For

proving a sequent $\Gamma \vdash \Delta$, we can, thus, assume all formulas in Γ and need to show one of the formulas in Δ, or at least show their disjunction.

This list Δ of alternatives to prove is simply preserved in the proof rules we saw so far:

$$(\wedge\text{R}) \quad \frac{\Gamma \vdash F, \Delta \quad \Gamma \vdash G, \Delta}{\Gamma \vdash F \wedge G, \Delta}$$

$$(\rightarrow\text{R}) \quad \frac{\Gamma, F \vdash G, \Delta}{\Gamma \vdash F \rightarrow G, \Delta}$$

$$(\vee\text{R}) \quad \frac{\Gamma \vdash F, G, \Delta}{\Gamma \vdash F \vee G, \Delta}$$

For example in rule $\wedge$R, the same succedent Δ is still available in both premises, because a proof of Δ from the assumptions Γ in either premise would also prove Δ from the assumptions Γ in the conclusion.

How do we prove a bisubjunction $P \leftrightarrow Q$? Going back to the semanatics, we see that bisubjunction is really like two implications $P \rightarrow Q$, $Q \rightarrow P$ joined with a conjunction. This gives us the rule $\leftrightarrow$R.

$$(\leftrightarrow\text{R}) \quad \frac{\Gamma \vdash F \rightarrow G, \Delta \quad \Gamma \vdash G \rightarrow F, \Delta}{\Gamma \vdash F \leftrightarrow G, \Delta}$$

Finally, we can prove a negation $\neg F$ by assuming the converse F and going for a contradiction. In fact, since we may have already gathered a number of other alternatives Δ to prove, all we need to do to prove $\neg F$ from a list of assumptions Γ with a list of alternatives Δ is to prove the remaining alternatives Δ from assuming Γ as well as the opposite F:

$$(\neg\text{R}) \quad \frac{\Gamma, F \vdash \Delta}{\Gamma \vdash \neg F, \Delta}$$

Does this list of rules handle all operators? There's one rule per operator, which is a good thing. The catch is that there's really only one rule per operator so far. If the operators occur on the right, so in the succedent, then the respective proof rules tell us what to do. But the implication proof rule $\rightarrow$R is good about pushing assumptions into the antecedent. What if it pushes a conjunction $F \wedge G$ into the antecedent? Is there a proof rule to handle what happens then?

Not yet. But there should be a rule for handling the case where there's a conjunction $F \wedge G$ among the list of assumptions in the antecedent. In fact, for every logical operator, there should be a right proof rule handling the case where it is the top-level operator on the right in the succedent as well as a left proof rule handling when it appears on the left in the antecedent.

5.3 Left Rules

When we find a conjunction $F \wedge G$ among the list of assumptions in the antecedent, then we can safely split it into two separate assumptions F as well as G:

$$(\wedge\text{L})\ \frac{\Gamma, F, G \vdash \Delta}{\Gamma, F \wedge G \vdash \Delta}$$

Proving a sequent that has a conjunction $F \wedge G$ among its assumptions in the antecedent is the same as proving it with two separate assumptions F as well as G instead.

What happens when we have a disjunction $F \vee G$ among our assumptions in the antecedent? In that case we have no way of knowing whether F or whether G is true. All we know is that either of them is. But we still succeed with a proof if we manage to show the sequent both when assuming F as well as when, instead, assuming G, because while either are possible, the assumption $F \vee G$ implies that one of those cases has to apply.

$$(\vee\text{L})\ \frac{\Gamma, F \vdash \Delta \quad \Gamma, G \vdash \Delta}{\Gamma, F \vee G \vdash \Delta}$$

When an implication $F \to G$ is among the assumptions in the antecedent, then we can make use of that assumption by showing its respective assumption F and can then assume G instead. If we can assume $F \to G$ and show F then we can assume G:

$$(\to\text{L})\ \frac{\Gamma \vdash F, \Delta \quad \Gamma, G \vdash \Delta}{\Gamma, F \to G \vdash \Delta}$$

Wait a moment. The left premise does not actually show F from the assumptions Γ, because it only shows the succedent F, Δ which is interpreted disjunctively. So it is possible that the left premise does not show F but merely Δ. But in that case, the conclusion is justified as well, because it also has the antecedent Δ as the list of alternatives to show.

We leave the left rule for the operator $\leftrightarrow$ as an exercise, so the only remaining case is to handle a negation $\neg F$ among the assumptions in the antecedent. If we assume $\neg F$ then it is also sufficient if we can show the opposite F (recall the semantics of sequents):

$$(\neg\text{L})\ \frac{\Gamma \vdash F, \Delta}{\Gamma, \neg F \vdash \Delta}$$

To understand, we can first pretend there would be no succedent Δ. What happens if there is no succedent? Then the empty disjunction that it means is equivalent to the formula $\bot$ that is never true in any interpretation. In that special case, rule $\neg$L says that for proving a contradiction $\bot$ from assumptions Δ and $\neg F$, it is sufficient to prove the opposite F from the remaining assumptions Γ.

5.4 Closing and Forking

The above proof rules excel at splitting operators off of propositional logical formulas. But they never actually prove anything on their own except simplifying all formulas until only atomic propositions are left. What is missing is the observation that a sequent can be proved easily when the same formula F is in the antecedent and succedent with the identity proof rule called id:

$$\text{(id)}\ \frac{}{\Gamma, F \vdash F, \Delta}$$

Whenever we find the same formula F in the antecedent and succedent, we can use rule id to prove that sequent without any further questions (no premise, i.e. no more remaining subgoals).

Another insightful proof rule is the cut proof rule, which enables us to first prove an arbitrary formula C on the left premise and then assume C on the right premise.

$$\text{(cut)}\ \frac{\Gamma \vdash C, \Delta \quad \Gamma, C \vdash \Delta}{\Gamma \vdash \Delta}$$

Think of C as a lemma that is proved in the left premise and then assumed to hold in the right premise. The twist is again that the left premise does not necessarily prove C but might also settle for proving another alternative in the remaining succedent Δ, but that also establishes the succedent Δ of the conclusion. The primary purpose of the cut rule is for ingenious theoretical studies of reasoning [Gen35] as well as to find clever shortcuts in practical proofs by first proving a lemma C that subsequently helps multiple times in the remaining proof. It plays a crucial role in constructive logics, too.

All these sequent calculus proof rules are *sound*, that is, if all their premises are valid, then their conclusion is valid. Especially if there are no premises any more because we were able to use the identity proof rule id on all premises, then the conclusion is valid, which is what we were hoping to achieve with a proof.

5.5 Conducting Sequent Calculus Proofs

As an example, let's prove formula (1). Sequent calculus proofs are conducted in a bit of a funny way by starting with the conjecture at the bottom

$$\vdash (p \to q) \leftrightarrow (\neg p \vee q)$$

and then working our way upwards by applying proof rules to the remaining sequents. The reason why we work like that is that in (sound!) sequent calculus proof rules validity of all premises implies validity of the conclusion. So if we start with our conjecture at the bottom and work our way upwards, then if we are able to prove all premises then the conclusion at the bottom will be valid, too. We apply sequent calculus rules from the bottom to the top but, when a proof is done, their soundness makes validity inherit from the top to the bottom.

$$(\wedge\text{R})\ \frac{\Gamma \vdash F, \Delta \quad \Gamma \vdash G, \Delta}{\Gamma \vdash F \wedge G, \Delta} \qquad (\wedge\text{L})\ \frac{\Gamma, F, G \vdash \Delta}{\Gamma, F \wedge G \vdash \Delta}$$

$$(\vee\text{R})\ \frac{\Gamma \vdash F, G, \Delta}{\Gamma \vdash F \vee G, \Delta} \qquad (\vee\text{L})\ \frac{\Gamma, F \vdash \Delta \quad \Gamma, G \vdash \Delta}{\Gamma, F \vee G \vdash \Delta}$$

$$(\to\text{R})\ \frac{\Gamma, F \vdash G, \Delta}{\Gamma \vdash F \to G, \Delta} \qquad (\to\text{L})\ \frac{\Gamma \vdash F, \Delta \quad \Gamma, G \vdash \Delta}{\Gamma, F \to G \vdash \Delta}$$

$$(\neg\text{R})\ \frac{\Gamma, F \vdash \Delta}{\Gamma \vdash \neg F, \Delta} \qquad (\neg\text{L})\ \frac{\Gamma \vdash F, \Delta}{\Gamma, \neg F \vdash \Delta}$$

$$(\text{id})\ \frac{}{\Gamma, F \vdash F, \Delta} \qquad (\text{cut})\ \frac{\Gamma \vdash C, \Delta \quad \Gamma, C \vdash \Delta}{\Gamma \vdash \Delta}$$

Figure 1: Sequent calculus proof rules for propositional logic

Enough said. Let's prove formula (1) in sequent calculus:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{*}{p \vdash p, q}\,\text{id} \quad \dfrac{*}{q, p \vdash q}\,\text{id}}{p \to q, p \vdash q}\,{\to}\text{L}}{p \to q \vdash \neg p, q}\,\neg\text{R}}{p \to q \vdash \neg p \vee q}\,\vee\text{R}}{\vdash (p \to q) \to \neg p \vee q}\,{\to}\text{R} \qquad \dfrac{\dfrac{\dfrac{\dfrac{*}{p \vdash p, q}\,\text{id}}{\neg p, p \vdash q}\,\neg\text{L} \quad \dfrac{*}{q, p \vdash q}\,\text{id}}{\neg p \vee q, p \vdash q}\,\vee\text{L}}{\neg p \vee q \vdash p \to q}\,{\to}\text{L}}{\vdash \neg p \vee q \to p \to q}\,{\to}\text{L}}{\vdash (p \to q) \leftrightarrow (\neg p \vee q)}\,{\leftrightarrow}\text{R}$$

6 Soundness

Having conducted a sequent calculus proof, the most pressing question is what a proof proves. Of course, as we already alluded to before, a proof in a sound deductive system implies the validity of the conclusion.

Definition 5 (Soundness of a proof rule)**.** A sequent calculus proof rule

$$\frac{\Gamma_1 \vdash \Delta_1 \quad \ldots \quad \Gamma_n \vdash \Delta_n}{\Gamma \vdash \Delta}$$

is *sound* iff the validity of all premises implies the validity of the conclusion:

$$\text{if } \vDash (\Gamma_1 \vdash \Delta_1) \text{ and } \ldots \text{ and } \vDash (\Gamma_n \vdash \Delta_n) \text{ then } \vDash (\Gamma \vdash \Delta)$$

Recall from Def. 4 that the meaning of the sequent $\Gamma \vdash \Delta$ is the same as that of the formula $\left(\bigwedge_{F \in \Gamma} F\right) \to \left(\bigvee_{G \in \Delta} G\right)$.

Lemma 6 (Soundness of propositional logic proof rules)**.** *All propositional logic proof rules (summarized again in Fig. 1), are sound.*

Proof. It is crucial to prove soundness for all proof rules. We will, nevertheless, only prove it for one rule and leave the others as exercises. But we will prove that rule with exceeding care.

∧R That proof rule ∧R is sound can be shown as follows. Assume that both of its premises $\Gamma \vdash F, \Delta$ and $\Gamma \vdash G, \Delta$ are valid, i.e. both $(\bigwedge_{F\in\Gamma} F) \to F \vee (\bigvee_{G\in\Delta} G)$ and $(\bigwedge_{F\in\Gamma} F) \to G \vee (\bigvee_{G\in\Delta} G)$ are true in all interpretations. We need to show that the conclusion $\Gamma \vdash F \wedge G, \Delta$ is then also valid, i.e. $\vDash (\Gamma \vdash F \wedge G, \Delta)$, which means that $(\bigwedge_{F\in\Gamma} F) \to (F \wedge G) \vee (\bigvee_{G\in\Delta} G)$ is true in all interpretations. Consider any interpretation I and show that $I \models (\bigwedge_{F\in\Gamma} F) \to (F \wedge G) \vee (\bigvee_{G\in\Delta} G)$. If any of the antecedent formulas $F \in \Gamma$ is false in I ($I \not\models F$) or any of the remaining succedent formulas $G \in \Delta$ is true ($I \models G$), then $I \models (\bigwedge_{F\in\Gamma} F) \to (F \wedge G) \vee (\bigvee_{G\in\Delta} G)$. Otherwise, all antecedent formulas in Γ are true $I \models \bigwedge_{F\in\Gamma} F$ and all Δ formulas are false $I \not\models \bigvee_{G\in\Gamma} G$.

By premise, $I \models (\bigwedge_{F\in\Gamma} F) \to F \vee (\bigvee_{G\in\Delta} G)$ and $I \models (\bigwedge_{F\in\Gamma} F) \to G \vee (\bigvee_{G\in\Delta} G)$. Since antecedents in Γ are true and succedents in Δ false in I, this implies $I \models F$ and $I \models G$. By Def. 2, these imply $I \models F \wedge G$, which implies that the conclusion is true in I, i.e. $I \models (\bigwedge_{F\in\Gamma} F) \to (F \wedge G) \vee (\bigvee_{G\in\Delta} G)$. □

In fact, the prelude of the soundness argument is common to all proof rules so that one usually just assumes right away without loss of generality that the common antecedent Γ is true while the common succedent Δ false in the current interpretation I.

Now that all proof rules of propositional logic are sound it is easy to see that the whole proof calculus is sound, because a proof is entirely built by applying sound proof rules so validity of all premises (of which there are none in a completed proof) implies validity of the conclusion. Because this is so important and we want to practice the important proof principle of induction, we will show this explicitly.

Theorem 7 (Soundness of propositional logic)**.** *The sequent calculus of propositional logic is sound, i.e. it only proves valid formulas. That is, if* $\vdash F$ *has a proof in the propositional sequent calculus, then* F *is valid, i.e.* $\vDash F$*.*

Proof. What we need to show is that if $\vdash F$ is the conclusion of a completed sequent calculus proof, then F is valid, i.e. $\vDash F$. A proof of the sequent $\vdash F$ will consist of proofs of sequents of the more general shape $\Gamma \vdash \Delta$. So we instead prove the more general statement that a proof of $\Gamma \vdash \Delta$ implies $\vDash (\Gamma \vdash \Delta)$. We will prove this by induction on the structure of the proof. That is, we will prove it for the smallest possible proofs. And then, assuming that the proofs of the smaller pieces of a proof have valid conclusions, we will show that one more proof step preserves validity.

1. The only proofs with just 1 proof step are of the form

$$\text{id}\,\frac{*}{\Gamma, F \vdash F, \Delta}$$

Its conclusion is valid, because assumption F in the antecedent trivially implies F in the succedent.

2. Consider any proof ending with a proof step of this form:

$$\frac{\Gamma_1 \vdash \Delta_1 \quad \ldots \quad \Gamma_n \vdash \Delta_n}{\Gamma \vdash \Delta} \tag{5}$$

   By induction hypothesis, we can assume that the (smaller!) proofs of the premises $\Gamma_1 \vdash \Delta_1$ and $\ldots \Gamma_n \vdash \Delta_n$ already imply the validity of their respective conclusions so $\vDash (\Gamma_1 \vdash \Delta_1)$ and $\ldots \vDash (\Gamma_n \vdash \Delta_n)$.

   The proof rule used in the proof step (5) must have been one of the proof rules of the sequent calculus of propositional logic. All these sequent calculus proof rules of propositional logic are sound by Lemma 6. Consequently, $\vDash (\Gamma \vdash \Delta)$, so the conclusion of the proof (5) is valid. □

Soundness is one thing, and most crucial for any correct reasoning. But since propositional logic is so simple, it enjoys other pleasant properties. It is also the case that every valid propositional logic formula will be provable from the sequent calculus proof rules in Fig. 1, which is called *completeness*.

Theorem 8 (Completeness of propositional logic)**.** *The sequent calculus of propositional logic is complete, i.e. it proves all valid formulas. That is, if F is valid, so $\vDash F$ then $\vdash F$ has a proof in the propositional sequent calculus.*

In fact, because propositional logic is so simple, it is perfectly decidable whether a propositional logical formula is valid. You already knew this, however, because we discussed an effective procedure for deciding propositional validity—truth tables.

Theorem 9 (Decidability of propositional logic)**.** *Propositional logic is* decidable*, i.e. there is an algorithm that accepts any propositional logical formula as input and correctly outputs "valid" or "not valid" in finite time.*

References

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. I. *Math. Zeit.*, 39(2):176–210, 1935.

[HHK+15] Chris Hawblitzel, Jon Howell, Manos Kapritsos, Jacob R. Lorch, Bryan Parno, Michael L. Roberts, Srinath Setty, and Brian Zill. Ironfleet: Proving practical distributed systems correct. In *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015.

[HHL+14] Chris Hawblitzel, Jon Howell, Jacob R. Lorch, Arjun Narayan, Bryan Parno, Danfeng Zhang, and Brian Zill. Ironclad apps: End-to-end security via automated full-system verification. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, 2014.