How To Do Proofs
(for 1B Semantics of Programming Languages)

Myra VanInwegen, Peter Sewell

February 9, 2004

The purpose of this handout is give a general guide as to how to prove theorems. This should give you some help in answering questions that begin with "Show that the following is true ...". It is based on notes by Myra VanInwegen, with additional text added by Peter Sewell.

The focus here is on doing *informal but rigorous proofs*. These are rather different from the *formal proofs*, in Natural Deduction or Sequent Calculus, that were introduced in the Logic and Proof course. Formal proofs are derivations in one of those proof systems – they are in a completely well-defined form, but are often far too verbose to deal with by hand (although they can be machine-checked). Informal proofs, on the other hand, are the usual mathematical notion of proof: written arguments to persuade the reader that you could, if pushed, write a fully formal proof.

This is important for two reasons. Most obviously, you should learn how to do these proofs. More subtly, but more importantly, only by working with the mathematical definitions in some way can you develop a good intuition for what they mean — trying to do some proofs is the best way of understanding the definitions.

1 How to go about it

Proofs differ, but for many of those you meet the following steps should be helpful.

1. Make sure the statement of the conjecture is precisely defined. In particular, make sure you understand any strange notation, and find the definitions of all the auxiliary gadgets involved (e.g. definitions of any typing or reduction relations mentioned in the statement, or any other predicates or functions).

2. Try to understand at an intuitive level what the conjecture is saying – verbalize out loud the basic point. For example, for a Type Preservation conjecture, the basic point might be something like "if a well-typed configuration reduces, the result is still well-typed (with the same type)".

3. Try to understand intuitively why it is true (or false...). Identify what the most interesting cases might be — the cases that you think are most likely to be suspicious, or hard to prove. Sometimes it's good to start with the easy cases (if the setting is unfamiliar to you); sometimes it's good to start with the hard cases (to find any interesting problems as soon as possible).

4. Think of a good basic strategy. This might be:

   (a) simple logic manipulations;

   (b) collecting together earlier results, again by simple logic; or

   (c) some kind of induction.

5. Try it! (remembering you might have to backtrack if you discover you picked a strategy that doesn't work well for this conjecture). This might involve any of the following:

   (a) Expanding definitions, inlining them. Sometimes you can just blindly expand all definitions, but more often it's important to expand only the definitions which you want to work with the internal structure of — otherwise things just get too verbose.

   (b) Making abbreviations — defining a new variable to stand for some complex gadget you're working with, saying e.g.

   ```
   where e = (let x:int = 7+2 in x+x)
   ```

   Take care with choosing variable names.

   (c) Doing *equational reasoning*, e.g.

   ```
   e = e1   by ...
     = e2   by ...
     = e3   as ...
   ```

   Here the `e` might be any mathematical object — arithmetic expressions, or expressions of some grammar, or formulae. Some handy equations over formulae are given in §2.2.

   (d) Proving a formula based on its structure. For example, to prove a formula $\forall x \in S.P(x)$ you would often assume you have an arbitrary x and then try to prove $P(x)$.

   Take an arbitrary $x \in S$.
   We now have to show $P(x)$:

   This is covered in detail in §2.3. Much proof is of this form, automatically driven by the structure of the formula.

   (e) Using an assumption you've made above.

   (f) Induction. As covered in the 1B Semantics notes, there are various kinds of induction you might want to use: mathematical induction over the natural numbers, structural induction over the elements of some grammar, or rule induction over the rules defining some relation (especially a reduction or typing relation). For each, you should:

      i. Decide (and state!) what kind of induction you're using. This may need some thought and experience, and you might have to backtrack.

      ii. Remind yourself what the induction principle is exactly.

      iii. Decide on the induction hypothesis you're going to use, writing down a predicate Φ which is such that the conclusion of the induction principle implies the thing you're trying to prove. Again, this might need some thought. Take care with the quantifiers here — it's suspicious if your definition of Φ has any globally-free variables...

      iv. Go through each of the premises of the induction principle and prove each one (using any of these techniques as appropriate). Many of those premises will be implications, e.g. $\forall x \in \mathbb{N}.\Phi(x) \Rightarrow \Phi(x+1)$, for which you can do a proof based on the structure of the formula — taking an arbitrary x, assuming $\Phi(x)$, and trying to prove $\Phi(x+1)$. Usually at some point in the latter you'd make use of the assumption $\Phi(x)$.

6. In all of the above, remember: the point of doing a proof on paper is to *use* the formalism to *help you think* — to help you cover all cases, precisely — and also to *communicate with the reader*. For both, you need to write clearly:

(a) Use enough words! "Assume", "We have to show", "By such-and-such we know", "Hence",...

(b) Don't use random squiggles. It's good to have formulae properly nested within text, with and no "$\Rightarrow$" or "$\because$" between lines of text.

7. If it hasn't worked yet... either

   (a) you've make some local mistake, e.g. mis-instantiated something, or used the same variable for two different things, or not noticed that you have a definition you should have expanded or an assumption you should have used. Fix it and continue.

   (b) you've discovered that the conjecture is really false. Usually at this point it's a good idea to construct a counterexample that is as simple as possible, and to check carefully that it really is a counterexample.

   (c) you need to try a different strategy — often, to use a different induction principle or to strengthen your induction hypothesis.

   (d) you didn't really understand intuitively what the conjecture is saying, or what the definitions it uses mean. Go back to them again.

8. If it has worked: read through it, skeptically, and check. Maybe you'll need to *re-write* it to make it comprehensible: proof *discovery* is not the same as proof *exposition*. See the example proofs in the Semantics notes.

9. Finally, give it to someone else, as skeptical and careful as you can find, to see if they believe it — to see if they believe that *what you've written down is a proof*, not that they believe that *the conjecture is true*.

2 And in More Detail...

First, I'll explain informal proof intuitively, giving a couple of examples. Then I'll explain how this intuition is reflected in the sequent rules from Logic and Proof.

In the following, I'll call any logic statement a formula. In general, what we'll be trying to do is *prove* a formula, using a collection of formulas that we know to be true or are assuming to be true. There's a big difference between *using* a formula and *proving* a formula. In fact, what you do is in many ways opposite. So, I'll start by explaining how to *prove* a formula.

2.1 Meet the Connectives

Here are the logical connectives and a very brief decription of what each means.

$P \wedge Q$	P and Q are both true
$P \vee Q$	P is true, or Q is true, or both are true
$\neg P$	P is not true (P is false)
$P \Rightarrow Q$	if P is true then Q is true
$P \Leftrightarrow Q$	P is true exactly when Q is true
$\forall x \in S.P(x)$	for all x in S, P is true of x
$\exists x \in S.P(x)$	there exists an x in S such that P holds of x

2.2 Equivalences

These are formulas that mean the same thing, and this is indicated by a $\simeq$ between them. The fact that they are equivalent to each other is justified by the truth tables of the connectives.

definition of $\Rightarrow$	$P \Rightarrow Q$	$\simeq$	$\neg P \vee Q$
definition of $\Leftrightarrow$	$P \Leftrightarrow Q$	$\simeq$	$(P \Rightarrow Q) \wedge (Q \Rightarrow P)$
definition of $\neg$	$\neg P$	$\simeq$	$P \Rightarrow \text{false}$
de Morgan's Laws	$\neg(P \wedge Q)$	$\simeq$	$\neg P \vee \neg Q$
	$\neg(P \vee Q)$	$\simeq$	$\neg P \wedge \neg Q$
extension to quantifiers	$\neg(\forall x.P(x))$	$\simeq$	$\exists x.\neg P(x)$
	$\neg(\exists x.P(x))$	$\simeq$	$\forall x.\neg P(x)$
distributive laws	$P \vee (Q \wedge R)$	$\simeq$	$(P \vee Q) \wedge (P \vee R)$
	$P \wedge (Q \vee R)$	$\simeq$	$(P \wedge Q) \vee (P \wedge R)$
coalescing quantifiers	$(\forall x.P(x)) \wedge (\forall x.Q(x))$	$\simeq$	$\forall x.(P(x) \wedge Q(x))$
	$(\exists x.P(x)) \vee (\exists x.Q(x))$	$\simeq$	$\exists x.(P(x) \vee Q(x))$
these ones apply if	$(\forall x.P(x)) \wedge Q$	$\simeq$	$(\forall x.P(x) \wedge Q)$
x is not free in Q	$(\forall x.P(x)) \vee Q$	$\simeq$	$(\forall x.P(x) \vee Q)$
	$(\exists x.P(x)) \wedge Q$	$\simeq$	$(\exists x.P(x) \wedge Q)$
	$(\exists x.P(x)) \vee Q$	$\simeq$	$(\exists x.P(x) \vee Q)$

2.3 How to Prove a Formula

For each of the logical connectives, I'll explain how to handle them.

$\boxed{\forall x \in S.P(x)}$ This means "For all x in S, P is true of x." Such a formula is called a universally quantified formula. The goal is to prove that the property P, which has some xs somewhere in it, is true no matter what value in S x takes on. Often the "$\in S$" is left out. For example, in a discussion of lists, you might be asked to prove $\forall l.\mathtt{length}\, l > 0 \Rightarrow \exists x.\mathtt{member}(x, l)$. Obviously, l is a list, even if it isn't explicitly stated as such.

There are several choices as to how to prove a formula beginning with $\forall x$. The standard thing to do is to just prove $P(x)$, not assuming anything about x. Thus, in doing the proof you sort of just mentally strip off the $\forall x$. What you would write when doing this is "Let x be any S". However, there are some subtleties—if you're already using an x for something else, you can't use the same x, because then you *would* be assuming something about x, namely that it equals the x you're already using. In this case, you need to use alpha-conversion[1] to change the formula you want to prove to $\forall y \in S.P(y)$, where y is some variable you're not already using, and then prove $P(y)$. What you could write in this case is "Since x is already in use, we'll prove the property of y".

An alternative is induction, if S is a set that is defined with a structural definition. Many objects you're likely to be proving properties of are defined with a structural definition. This includes natural numbers, lists, trees, and terms of a computer language. Sometimes you can use induction over the natural numbers to prove things about other objects, such as graphs, by inducting over the number of nodes (or edges) in a graph.

You use induction when you see that during the course of the proof you would need to use the property P for the subparts of x in order to prove it for x. This usually ends up being the case if P involves functions defined recursively (i.e., the return value for the function depends on the function value on the subparts of the argument).

A special case of induction is case analysis. It's basically induction where you don't use the inductive hypothesis: you just prove the property for each possible form that x could have. Case analysis can be used to prove the theorem about lists above.

A final possibility (which you can use for all formulas, not just for universally quantified ones) is to assume the contrary, and then derive a contradiction.

$\boxed{\exists x \in S.P(x)}$ This says "There exists an x in S such that P holds of x." Such a formula is called an existentially quantified formula. The main way to prove this is to figure out what x has to be (that is, to find a concrete representation of it), and then prove that P holds of that value. Sometimes you can't give a completely specified value, since the value you pick for x has to depend on the values of other things you have floating around. For example, say you want to prove

$$\forall x, y \in \Re . x < y \wedge \sin x < 0 \wedge \sin y > 0 \Rightarrow \exists z . x < z \wedge z < y \wedge \sin z = 0$$

where $\Re$ is the set of real numbers. By the time you get to dealing with the $\exists z . x < z \wedge z < y \wedge \sin z = 0$, you will have already assumed that x and y were any real numbers. Thus the value you choose for z has to depend on whatever x and y are.

An alternative way to prove $\exists x \in S.P(x)$ is, of course, to assume that no such x exists, and derive a contradiction.

To summarize what I've gone over so far: to *prove* a universally quantified formula, you must prove it for a generic variable, one that you haven't used before. To prove an existentially quantified formula, you get to choose a value that you want to prove the property of.

$\boxed{P \Rightarrow Q}$ This says "If P is true, then Q is true". Such a formula is called an implication, and it is often pronounced "P implies Q". The part before the $\Rightarrow$ sign (here P) is called the antecedent, and the part after the $\Rightarrow$ sign (here Q) is called the consequent. $P \Rightarrow Q$ is equivalent to $\neg P \vee Q$, and so if P is false, or if Q is true, then $P \Rightarrow Q$ is true.

The standard way to prove this is to assume P, then use it to help you prove Q. Note that I said that you will be *using* P. Thus you will need to follow the rules in Section 2.4 to deal with the logical connectives in P.

Other ways to prove $P \Rightarrow Q$ involve the fact that it is equivalent to $\neg P \vee Q$. Thus, you can prove $\neg P$ without bothering with Q, or you can just prove Q without bothering with P. To reason by contradiction you assume that P is true and that Q is not true, and derive a contradiction.

Another alternative is to prove the contrapositive: $\neg Q \Rightarrow \neg P$, which is equivalent to it.

[1] Alpha-equivalence says that the name of a bound variable doesn't matter, so you can change it at will (this is called alpha-conversion). You'll get to know the exact meaning of this soon enough so I won't explain this here.

$\boxed{P \Leftrightarrow Q}$ This says "P is true if and only if Q is true". The phrase "if and only if" is usually abbreviated "iff". Basically, this means that P and Q are either both true, or both false.

Iff is usually used in two main ways: one is where the equivalence is due to one formula being a definition of another. For example, $A \subseteq B \Leftrightarrow (\forall x.\, x \in A \Rightarrow x \in B)$ is the standard definition of subset. For these iff statements, you don't have to prove them. The other use of iff is to state the equivalence of two different things. For example, you could define an SML function `fact`:

```
fun fact 0 = 1
  | fact n = n * fact (n - 1)
```

Since in SML whole numbers are integers (both positive and negative) you may be asked to prove: $\texttt{fact}\ x$ terminates $\Leftrightarrow x \geq 0$. The standard way to do this is us the equivalence $P \Leftrightarrow Q$ is equivalent to $P \Rightarrow Q \wedge Q \Rightarrow P$. And so you'd prove that $(\texttt{fact}\ x \text{ terminates} \Rightarrow x \geq 0) \wedge (x \geq 0 \Rightarrow \texttt{fact}\ x \text{ terminates})$.

$\boxed{\neg P}$ This says "P is not true". It is equivalent to $P \Rightarrow$ false, thus this is one of the ways you prove it: you assume that P is true, and derive a contradiction (that is, you prove false). Here's an example of this, which you'll run into later this year: the undecidability of the halting problem can be rephrased as $\neg\exists x \in RM.\, x$ *solves the halting problem*, where RM is the set of register machines. The proof of this in your Computation Theory notes follows exactly the pattern I described—it assumes there is such a machine and derives a contradiction.

The other major way to prove $\neg P$ is to figure out what the negation of P is, using equivalences like De Morgan's Law, and then prove that. For example, to prove $\neg\forall x \in \mathcal{N}.\, \exists y \in \mathcal{N}.\, x = y^2$, where $\mathcal{N}$ is the set of natural numbers, you could push in the negation to get: $\exists x \in \mathcal{N}.\, \forall y \in \mathcal{N}.\, x \neq y^2$, and then you could prove that.

$\boxed{P \wedge Q}$ This says "P is true and Q is true". Such a formula is called a conjunction. To prove this, you have to prove P, and you have to prove Q.

$\boxed{P \vee Q}$ This says "P is true or Q is true". This is *inclusive* or: if P and Q are both true, then $P \vee Q$ is still true. Such a formula is called a disjunction. To prove this, you can prove P or you can prove Q. You have to choose which one to prove. For example, if you need to prove $(5 \bmod 2 = 0) \vee (5 \bmod 2 = 1)$, then you'll choose the second one and prove that.

However, as with existentials, the choice of which one to prove will often depend on the values of other things, like universally quantified variables. For example, when you are studying the theory of programming languages (you will get a bit of this in Semantics), you might be asked to prove

$$\begin{aligned}\forall P \in ML.\quad & P \textit{ is properly typed} \Rightarrow \\ & (\textit{the evaluation of } P \textit{ runs forever}) \vee (P \textit{ evaluates to a value})\end{aligned}$$

where ML is the set of all ML programs. You don't know in advance which of these will be the case, since some programs do run forever, and some do evaluate to a value. Generally, the best way to prove the disjunction in this case (when you don't know in advance which will hold) is to use the equivalence with implication. For example, you can use the fact that $P \vee Q$ is equivalent to $\neg P \Rightarrow Q$, then assume $\neg P$, then use this to prove Q. For example, your best bet to proving this programming languages theorem is to assume that the evaluation of P doesn't run forever, and use this to prove that P evaluates to a value.

2.4 How to Use a Formula

You often end up using a formula to prove other formulas. You can use a formula if someone has already proved that it's true, or you are assuming it because it was in an implication, namely, the A in $A \Rightarrow B$. For each logical connective, I'll tell you how to use it.

$\boxed{\forall x \in S. P(x)}$ This formula says that something is true of *all* elements of S. Thus, when you use it, you can pick any value at all to use instead of x (call it v), and then you can use $P(v)$.

$\boxed{\exists x \in S. P(x)}$ This formula says that there is some x that satisfies P. However, you do not know what it is, so you can not assume anything about it. The usual approach it to just say that the thing that is being said to exist is just x, and use the fact that P holds of x to prove something else. However, if you're already using an x for something else, you have to pick another variable to represent the thing that exists.

To summarize this: to *use* a universally quantified formula, you can choose any value, and use that the formula holds for that variable. To prove an existentially quantified formula, you must not assume anything about the value that is said to exists, so you just use a variable (one that you haven't used before) to represent it. Note that this is more or less opposite of what you do when you prove a universally or existentially quantified formula.

$\boxed{\neg P}$ Usually, the main use of this formula is to prove the negation of something else. An example is the use of reduction to prove the unsolvability of various problems in the Computation Theory (you'll learn all about this in Lent term). You want to prove $\neg Q$, where Q states that a certain problem (Problem 1) is decidable (in other words, you want to prove that Problem 1 is not decidable). You know $\neg P$, where P states that another problem (Problem 2) is decidable (i.e. $\neg P$ says that Problem 2 is not decidable). What you do basically is this. You first prove $Q \Rightarrow P$, which says that if Problem 1 is decidable, then so is Problem 2. Since $Q \Rightarrow P \simeq \neg P \Rightarrow \neg Q$, you have now proved $\neg P \Rightarrow \neg Q$. You already know $\neg P$, so you use modus ponens[2] to get that $\neg Q$.

$\boxed{P \Rightarrow Q}$ The main way to use this is that you prove P, and then you use modus ponens to get Q, which you can then use.

$\boxed{P \Leftrightarrow Q}$ The main use of this is to replace an occurrence of P in a formula with Q, and vise versa.

$\boxed{P \wedge Q}$ Here you can use both P and Q. Note, you're not *required* to use both of them, but they are both true and are waiting to be used by you if you need them.

$\boxed{P \vee Q}$ Here, you know that one of P or Q is true, but you do not know which one. To use this to prove something else, you have to do a split: first you prove the thing using P, then you prove it using Q.

Note that in each of the above, there is again a difference in the way you use a formula, verses the way you prove it. They are in a way almost opposites. For example, in proving $P \wedge Q$, you have to prove both P and Q, but when you are using the formula, you don't have to use both of them.

3 An Example

There are several exercises in the Semantics notes that ask you to prove something. Here, we'll go back to Regular Languages and Finite Automata. (If they've faded, it's time to remind yourself of them.) The Pumping Lemma for regular sets (PL for short) is an astonishingly good example of the use of quantifiers. We'll go over the proof and use of the PL, paying special attention to the logic of what's happening.

3.1 Proving the PL

My favorite book on regular languages, finite automata, and their friends is the Hopcroft and Ullman book *Introduction to Automata Theory, Languages, and Computation*. You

[2]Modus ponens says that if $A \Rightarrow B$ and A are both true, then B is true.

should locate this book in your college library, and if it isn't there, insist that your DoS order it for you.

In the *Automata Theory* book, the Pumping Lemma is stated as: "Let L be a regular set. Then there is a constant n such that if z is any word in L, and $|z| \geq n$, we may write $z = uvw$ in such a way that $|uv| \leq n$, $|v| \geq 1$, and for all $i \geq 0$, uv^iw is in L." The Pumping Lemma is, in my experience, one of the most difficult things about learning automata theory. It is difficult because people don't know what to do with all those logical connectives. Let's write it as a logical formula.

$$\begin{array}{l} \forall L \in \mathit{RegularLanguages}. \\ \exists n.\ \forall z \in L.\ |z| \geq n \Rightarrow \\ \ \exists u\,v\,w.\ z = uvw \wedge |uv| \leq n \wedge |v| \geq 1 \wedge \\ \quad \forall i \geq 0.\ uv^iw \in L \end{array}$$

Complicated, eh? Well, let's prove it, using the facts that Hopcroft and Ullman have established in the chapters previous to the one wih the PL. I'll give the proof and put in square brackets comments about what I'm doing.

Let L be any regular language. [Here I'm dealing with the $\forall L \in \mathit{RegularLanguages}$ by stating that I'm not assuming anything about L.] Let M be a minimal-state deterministic finite state machine accepting L. [Here I'm *using* a fact that Hopcroft and Ullman have already proved about the equivalence of regular languages and finite automata.] Let n be the number of states in this finite state machine. [I'm dealing with the $\exists n$ by giving a very specific value of what it will be, based on the arbitrary L.] Let z be any word in L. [Thus I deal with $\forall z \in L$.] Assume that $|z| \geq n$. [Thus I'm taking care of the $\Rightarrow$ by assuming the antecedent.]

Say z is written $a_1a_2 \ldots a_m$, where $m \geq n$. Consider the states that M is in during the processing of the first n symbols of z, $a_1a_2 \ldots a_n$. There are $n+1$ of these states. Since there are only n states in M, there must be a duplicate. Say that after symbols a_j and a_k we are in the same state, state s (i.e. there's a loop from this state that the machine goes through as it accepts z), and say that $j < k$. Now, let $u = a_1a_2 \ldots a_j$. This represents the part of the string that gets you to state s the first time. Let $v = a_{j+1} \ldots a_k$. This represents the loop that takes you from s and back to it again. Let $w = a_{k+1} \ldots a_m$, the rest of word z. [We have chosen definite values for u, v, and w.] Then clearly $z = uvw$, since u, v, and w are just different sections of z. $|uv| \leq n$ since u and v occur within the first n symbols of z. $|v| \geq 1$ since $j < k$. [Note that we're dealing with the formulas connected with $\wedge$ by proving each of them.]

Now, let i be a natural number (i.e. ≥ 0). [This deals with $\forall i \geq 0$.] Then $uv^iw \in L$. [Finally our conclusion, but we have to explain why this is true.] This is because we can repeat the loop from s to s (represented by v) as many times as we like, and the resulting word will still be accepted by M.

3.2 Using the PL

Now we use the PL to prove that a language is not regular. This is a rewording of Example 3.1 from Hopcroft and Ullman. I'll show that $L = \{0^{i^2} | i \text{ is an integer}, i \geq 1\}$ is not regular. Note that L consists of all strings of 0's whose length is a perfect square. I will *use* the PL. I want to prove that L is not regular. I'll assume the negation (i.e., that L is regular) and derive a contradiction. So here we go. Remember that what I'm emphasizing here is not the finite automata stuff itself, but how to use a complicated theorem to prove something else.

Assume L is regular. We will use the PL to get a contradiction. Since L is regular, the PL applies to it. [We note that we're using the $\forall$ part of the PL for this particular L.] Let n be as described in the PL. [This takes care of using the $\exists n$. Note that we *are not* assuming anything about its actual value, just that it's a natural number.] Let $z = 0^{n^2}$. [Since the PL says that something is true of *all* zs, we can choose the one we want to use it for.] So by the

PL there exist u, v, and w such that $z = uvw$, $|uv| \leq n$, $|v| \geq 1$. [Note that we don't assume anything about what the u, v, and w actually are; the only thing we know about them is what the PL tells us about them. This is where people trying to use the PL usually screw up.] The PL then says that for any i, then $uv^iw \in L$. Well, then $uv^2w \in L$. [This is using the $\forall i \geq 0$ bit.] However, $n^2 < |uv^2w| \leq n^2 + n$, since $1 \leq |v| \leq n$. But $n^2 + n < (n+1)^2$. Thus $|uv^2w|$ lies properly between n^2 and $(n+1)^2$ and is thus not a perfect square. Thus uv^2w is not in L. This is a contradiction. Thus our assumption (that L was regular) was incorrect. Thus L is not a regular language.

4 Sequent Calculus Rules

In this section, I will show how the intuitive approach to things that I've described above is reflected in the sequent calculus rules. A sequent is $\Gamma \vdash \Delta$, where Γ and Δ are sets of formulas.[3] Technically, this means that

$$A_1 \wedge A_2 \wedge \ldots A_n \Rightarrow B_1 \vee B_2 \vee \ldots B_m \tag{1}$$

where $A_1, A_2, \ldots A_n$ are the formulas in Γ , and $B_1, B_2, \ldots B_n$ are the formulas in Δ. Less formally, this means "using the formulas in Γ we can prove that one of the formula in Δ is true." This is just the intuition I described above about using vs proving formulas, except that I only talked about proving that one formula is true, rather than proving that one of several formulas is true. In order to handle the $\vee$ connective, there can be any number of formulas on the right hand side of the $\vdash$.

For each logic connective,[4] I'll give the rules for it, and explain how it relates to the intuitive way of using or proving formulas. For each connective there are at least two rules for it: one for the left side of the $\vdash$, and one for the right side. This corresponds to having different ways to treat a formula depending on whether you're using it (for formulas on the left hand side of the $\vdash$) or proving it (for formulas on the right side of the $\vdash$).

It's easiest to understand these rules from the bottom up. The conclusion of the rule (the sequent below the horizontal line) is what we want to prove. The hypotheses of the rule (the sequents above the horizontal line) are how we go about proving it. We'll have to use more rules, adding to the top, to build up the proof of the hypothesis, but this at least tells us how to get going.

You can stop when the formula you have on the top is a *basic sequent*. This is $\Gamma \vdash \Delta$ where there's at least one formula (say P) that's in both Γ and Δ. You can see why this is the basic true formula: it says that if P and the other formulas in Γ are true, then P or one of the other formula in Δ is true.

In building proofs from these rules, there are several ways that you end up with formulas to the left of the $\vdash$, where you can use them rather than proving them. One is that you've already proved it before. This is shown with the cut rule:

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

The Δ, P in the first sequent in the hypotheses means that to the right of the $\vdash$ we have the set consisting of the formula P plus all the formulas in Δ, i.e., if all formulas in Γ are true, then P or one of the formulas in Δ is true. Similarly P, Γ to the left of the $\vdash$ in the second sequent means the set consisting of the formula P plus all the formulas in Γ.

We read this rule from the bottom up to make sense of it. Say we want to prove one of the formulas in Δ from the formulas in Γ, and we want to make use of a formula P that we've

[3]In your Logic and Proof notes, the symbol that divides Γ from Δ is $\Rightarrow$. However, that conflicts with the use of $\Rightarrow$ as implication. Thus I will use $\vdash$. You will see something similar in Semantics, where it separates assumptions (of the types of variables) from something that they allow you to prove.

[4]I won't mention iff here: as $P \Leftrightarrow Q$ is equivalent to $P \Rightarrow Q \wedge Q \Rightarrow P$, we don't need separate rules for it.

already proved. The fact that we've proved P is shown by the left hypothesis (of course, unless the left hypothesis is itself a basic sequent, then in a completed proof there will be more lines on top of the left hypothesis, showing the actual proof of the sequent). The fact that we are allowed to use P in the proof of Δ is shown in the right hand hypothesis. We continue to build the proof up from there, using P.

Some other ways of getting formulas to the left of the $\vdash$ are shown in the rules $(\neg r)$ and $(\Rightarrow r)$ below.

$\boxed{\forall x \in S.P(x)}$ The two rules for universally quantified formulas are:

$$\frac{P(v), \Gamma \vdash \Delta}{\forall x.P(x), \Gamma \vdash \Delta}\ (\forall l) \qquad \frac{\Gamma \vdash \Delta, P(x)}{\Gamma \vdash \Delta, \forall x.P(x)}\ (\forall r)$$

In the $(\forall r)$ rule, x must not be free in the conclusion.

Now, what's going on here? In the $(\forall l)$ rule, the $\forall x.P(x)$ is on the left side of the $\vdash$. Thus, we are using it (along with some other formula, those in Γ) to prove something (Δ). According to the intuition above, in order to *use* $\forall x.P(x)$, you can use it with any value, where v is used to represent that value. In the hypothesis, you see the formula $P(v)$ to the left of the $\vdash$. This is just P with v substituted for x. The use of this corresponds exactly to using the fact that P is true of any value whatsoever, since we are using it with v, which is any value of our choice.

In the $(\forall r)$ rule, the $\forall x.P(x)$ is on the right side of the $\vdash$. Thus, we are proving it. Thus, we need to prove it for a generic x. This is why the $\forall x$ is gone in the hypothesis. The x is still sitting somewhere in the P, but we're just using it as a plain variable, not assuming anything about it. And this explains the side condition too: "In the $(\forall r)$ rule, x must not be free in the conclusion." If x is not free in the conclusion, this means that x is not free in the formulas in Γ or Δ. That means the only place the x occurs free in the hypothesis is in P itself. This corresponds exactly with the requirement that we're proving that P is true of a generic x: if x were free in Γ or Δ, we *would* be assuming something about x, namely that value of x is the same as the x used in those formulas.

Note that induction is not mentioned in the rules. This is because the sequent calculus used here just deals with pure logic. In more complicated presentations of logic, it is explained how to define new types via structural induction, and from there you get mechanisms to allow you to do induction.

$\boxed{\exists x \in S.P(x)}$ The two rules for existentially quantified formulas are:

$$\frac{P(x), \Gamma \vdash \Delta}{\exists x.P(x), \Gamma \vdash \Delta}\ (\exists l) \qquad \frac{\Gamma \vdash \Delta, P(v)}{\Gamma \vdash \Delta, \exists x.P(x)}\ (\exists r)$$

In the $(\exists l)$ rule, x must not be free in the conclusion.

In $(\exists l)$, we are using $\exists x.P(x)$. Thus we cannot assume anything about the value that the formula says exists, so we just use it as x in the hypothesis. The side condition about x not being free in the conclusions comes from the requirement not to assume anything about x (since we don't know what it is). If x isn't free in the conclusion, then it's not free in Γ or Δ. If it were free in Γ or Δ, then we would be assuming that the x used there is the same as the x we're assuming exists, and this isn't allowed.

In $(\exists r)$, we are proving $\exists x.P(x)$. Thus we must pick a particular value (call it v) and prove P for that value. The value v is allowed to contain variables that are free in Γ or Δ, since you can set it to anything you want.

$\boxed{\neg P}$ The rules for negation are:

$$\frac{\Gamma \vdash \Delta, P}{\neg P, \Gamma \vdash \Delta}\ (\neg l) \qquad \frac{P, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg P}\ (\neg r)$$

Let's start with the right rule first. I said that the way to prove $\neg P$ is to assume P and derive a contradiction. If Δ is the empty set, then this is exactly what this rule says: If there are no formulas to the right hand side of the $\vdash$, then this means that the formulas in Γ are inconsistent (that means, they cannot all be true at the same time). This means that you have derived a contradiction. So if Δ is the empty set, the hypothesis of the rule says that, assuming P, you have obtained a contradiction. Thus, if you are absolutely certain about all your other hypotheses, then you can be sure that P is not true. The best way to understand the rule if Δ is not empty is to write out the meaning of the sequents in terms of the meaning of the sequent given by Equation 1 and work out the equivalence of the top and bottom of the rule using the equivalences in your Logic and Proof notes.

The easiest way to understand $(\neg l)$ is again by using equivalences.

$\boxed{P \Rightarrow Q}$ The two rules for implication are:

$$\frac{\Gamma \vdash \Delta, P \quad Q, \Gamma \vdash \Delta}{P \Rightarrow Q, \Gamma \vdash \Delta}\ (\Rightarrow l) \qquad \frac{P, \Gamma \vdash \Delta, Q}{\Gamma \vdash \Delta, P \Rightarrow Q}\ (\Rightarrow r)$$

The rule $(\Rightarrow l)$ easily understood using the intuitive explanation of how to use $P \Rightarrow Q$ given above. First, we have to prove P. This is the left hypothesis. Then we can use Q, which is what the right hypothesis says.

The right rule $(\Rightarrow r)$ is also easily understood. In order to prove $P \Rightarrow Q$, we assume P, then use this to prove Q. This is exactly what the hypothesis says.

$\boxed{P \wedge Q}$ The rules for conjunction are:

$$\frac{P, Q, \Gamma \vdash \Delta}{P \wedge Q, \Gamma \vdash \Delta}\ (\wedge l) \qquad \frac{\Gamma \vdash \Delta, P \quad \Gamma \vdash \Delta, Q}{\Gamma \vdash \Delta, P \wedge Q}\ (\wedge r)$$

Both of these rules are easily explained by the intuition above. The left rule $(\wedge l)$ says that when you use $P \wedge Q$, you can use P and Q. The right rule says that to prove $P \wedge Q$ you must prove P, and you must prove Q. You may wonder why we need separate hypotheses for the two different proofs. We can't just put P, Q to the right of the $\vdash$ in a single hypothesis, because that would mean that we're proving one of the other of them (see the meaning of the sequent given in Equation 1). So we need separate hypotheses to make sure that each of P and Q has actually been proved.

$\boxed{P \vee Q}$ The rules for disjunction are:

$$\frac{P, \Gamma \vdash \Delta \quad Q, \Gamma \vdash \Delta}{P \vee Q, \Gamma \vdash \Delta}\ (\vee l) \qquad \frac{\Gamma \vdash \Delta, P, Q}{\Gamma \vdash \Delta, P \vee Q}\ (\vee r)$$

These are also easily understood by the intuitive explanations above. The left rule says that to prove something (namely, one of the formulas in Δ) using $P \vee Q$, you need to prove it using P, then prove it using Q. The right rule says that in order to prove $P \vee Q$, you can prove one or the other. The hypothesis says that you can prove one or the other, because in order to show a sequent $\Gamma \vdash \Delta$ true, you only need to show that *one* of the formulas in Δ is true.

Acknowledgements

Many thanks to Myra for making her original notes available.