The semantics of propositional logic

Readings: Sections 1.3 and 1.4 of Huth and Ryan.

In this module, we will nail down the formal definition of a logical formula, and describe the semantics of propositional logic, which will be familiar to you from Math 135 and CS 251. Our final goal is to prove the soundness and completeness of propositional logic, but to do that, we need to be precise about induction, which we review carefully.

The formal language of formulas (1.3)

Our formulas can be viewed as strings over an alphabet composed of our atoms ($p, q, r, p_1, p_2, \ldots$), plus the symbols $\neg$, $\wedge$, $\vee$, $\rightarrow$, and the open and close parentheses, (and). ($\perp$ is a convenience in our proofs, and shouldn't appear in formulas.)

In the previous module we gave a grammar for the well-formed formulas of propositional logic:

$$\phi \quad ::= \quad p \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \rightarrow \phi$$

Recall that p ranges over all atoms and ϕ ranges over all **well-formed formulas** (wffs) (that is, formulas that conform to the grammar).

Also recall the notions of abstract syntax and operator precedence that we discussed in Module 2, and our use of parentheses to explicitly indicate precedence when necessary.

Using notions from CS 135 and CS 241, we can consider the parse tree (a rooted, ordered tree of fanout at most two) for the formula $((\neg p) \rightarrow (q \wedge r))$.

It's tempting to try to make this the definition of a formula, especially as you have seen how to prove things about trees in CS 134 and CS 136. But a tree also has a recursive definition, and the drawing is just a visualization of it. The tree is really representing the process of showing that the formula meets our first definition:

p,q,r are wffs because they are atoms.
$(q \wedge r)$ is a wff because q and r are wffs.
$(\neg p)$ is a wff because p is.
$((\neg p) \rightarrow (q \wedge r))$ is a wff because $(\neg p)$ and $(q \wedge r)$ are.

The formulas occurring in this process are **subformulas** of $((\neg p) \rightarrow (q \wedge r))$. They correspond to complete subtrees rooted at nodes of the parse tree.

Apart from reordering some of the lines in the justification on the previous slide, there is no other way to show that our formula meets the definition. Another way of saying this is that the parse tree is unique.

This is true for every formula that satisfies the definition. This will be important for our semantics. The textbook calls this the **inversion principle**, meaning it is possible to invert the process of building formulas.

The textbook asks you to take this on faith, but for those of little faith, we will prove it in a little while, since we'll have the tools to do so.

The ideas of a formal definition of a logical formula and a separate semantics for assigning a truth value to such a formula come from the work of Gottlob Frege in 1879, though the notation he used was quite different.

Many mathematicians and philosophers built on Frege's ideas, including Bertrand Russell, Alfred North Whitehead, Ludwig Wittgenstein, and Emil Post. Throughout their work, the notation changed and evolved.

What we have presented here is now standard terminology, though you will still see variations on it in other books (for example, the use of $\supset$ in place of $\rightarrow$).

The meaning of logical connectives

Recall our notion of semantic truth from Module 1. We have a set $\mathbb{B}$ of "boolean values" T and F, denoting truth and falsehood, respectively.

An **interpretation** (or **valuation** or **model**) of a formula ϕ is a mapping Φ of each propositional atom of ϕ onto a truth value.

Our example formula $((\neg p) \rightarrow (q \wedge r))$ has three atoms. If we assign T to p, T to q, and F to r, that is one interpretation. There are clearly $2^3 = 8$ different interpretations for this formula.

This should be quite familiar, but we are doing it carefully to be able to build on it. (Though we'll mostly use "interpretation" in this section, remember that word "model"; we'll see it a lot later on.)

In order to extend Φ to a map from entire formulas to $\mathbb{B}$, we need to establish meanings for the logical connectives. We do this below:

$$\text{meaning}(\wedge)(x, y) = \begin{cases} T & \text{if } x = y = T \\ F & \text{otherwise} \end{cases}$$

In words, the meaning of the conjunction operator is a function on two variables x and y, whose value is T precisely when x and y are both T.

$$\text{meaning}(\vee)(x, y) = \begin{cases} F & \text{if } x = y = F \\ T & \text{otherwise} \end{cases}$$

The meaning of the disjunction operator is a function on variables x and y, whose value is F precisely when x and y are both F.

$$\text{meaning}(\rightarrow)(x, y) = \begin{cases} F & \text{if } x = T \text{ and } y = F \\ T & \text{otherwise} \end{cases}$$

The meaning of the disjunction operator is a function on variables x and y, whose value is F precisely when $x = T$ and $y = F$.

$$\text{meaning}(\neg)(x) = \begin{cases} T & \text{if } x = F \\ F & \text{otherwise} \end{cases}$$

The meaning of the negation operator is a function on a single variable x, whose value is T when $x = F$, and F when $x = T$.

We then extend Φ to an interpretation function over all wffs as follows:

For each binary connective $\Box$, we have

$$\Phi(\phi \Box \psi) = \mathrm{meaning}(\Box)(\Phi(\phi), \Phi(\psi)) \ .$$

For the unary negation connective $\neg$, we have

$$\Phi(\neg \phi) = \mathrm{meaning}(\neg)(\Phi(\phi)) \ .$$

Returning to our example, $\neg p \rightarrow \big(q \wedge r\big)$, suppose that our interpretation function Φ is such that, as before, $\Phi\big(p\big) = T$, $\Phi\big(q\big) = T$, and $\Phi\big(r\big) = F$. Then we have

$$\begin{aligned}
\Phi(\neg p \rightarrow (q \wedge r)) &= \mathrm{meaning}(\rightarrow)(\Phi(\neg p), \Phi(q \wedge r)) \\
&= \mathrm{meaning}(\rightarrow)(\mathrm{meaning}(\neg)(\Phi(p)), \mathrm{meaning}(\wedge)(\Phi(q), \Phi(r))) \\
&= \mathrm{meaning}(\rightarrow)(\mathrm{meaning}(\neg)(T), \mathrm{meaning}(\wedge)(T, F)) \\
&= \mathrm{meaning}(\rightarrow)(F, F) \\
&= T
\end{aligned}$$

If all interpretation functions yield T when applied to a formula ϕ, then we say that ϕ is semantically true.

In summary:

If a formula is of the form $(\phi \wedge \psi)$, then it is assigned T if both ϕ and ψ are; otherwise it is assigned F.

If a formula is of the form $(\phi \vee \psi)$, then it is assigned F if both ϕ and ψ are; otherwise it is assigned T.

If a formula is of the form $(\phi \rightarrow \psi)$, then it is assigned F if ϕ is assigned T and ψ is assigned F; otherwise, it is assigned T.

If a formula is of the form $(\neg\phi)$, then it is assigned F if ϕ is assigned T; otherwise, it is assigned T.

We can iterate over all possible interpretations of a propositional formula using a device called a **truth table**. The truth tables depicted below describe explicitly the meanings of the connectives $\wedge$, $\vee$, and $\neg$ as functions on $\mathbb{B}$:

ϕ	ψ	$\phi \wedge \psi$
T	T	T
T	F	F
F	T	F
F	F	F

ϕ	ψ	$\phi \vee \psi$
T	T	T
T	F	T
F	T	T
F	F	F

ϕ	$\neg\phi$
T	F
F	T

These tables are just another way of describing the assignments on the previous slide.

We can also build a truth table for any formula – for example, $((\neg p) \rightarrow (q \wedge r))$.

p	q	r	$(\neg p)$	$(q \wedge r)$	$((\neg p) \rightarrow (q \wedge r))$
T	T	T	F	T	T
T	T	F	F	F	T
T	F	T	F	F	T
T	F	F	F	F	T
F	T	T	T	T	T
F	T	F	T	F	F
F	F	T	T	F	F
F	F	F	T	F	F

We read a truth table as follows: each row represents a possible interpretation function Φ. In the leftmost columns, we find the atomic statements. If there are n atomic statements, then the truth table will have 2^n rows, as there would then be 2^n possible truth assignments to those atomic statements, and therefore 2^n possible interpretation functions Φ.

Moving to the right, we obtain the values in each column by applying the meaning functions for the logical connectives to the truth values in the previous columns.

The last column denotes the truth assignment given by the interpretation function to the entire formula. If all values in the last column are T, then all interpretation functions map the formula to T, and we say that the formula is semantically true.

When we proved a sequent $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$, the idea was that eventually we should be able to show that if all of $\phi_1, \phi_2, \ldots \phi_n$ were true, then ψ would also be true. Now we have a way of describing a circumstance (a valuation) under which we can evaluate the truth value of formulas.

This suggests that what we want to show is that any valuation which makes $\phi_1, \phi_2, \ldots \phi_n$ true should make ψ true.

If this is in fact the case, we write $\phi_1, \phi_2, \ldots \phi_n \models \psi$. (The symbol $\models$ is read “models”).

As we said in Module 1, the central question to be answered is, what is the relationship between the two relations $\models$ and $\vdash$?

We define $\phi_1, \phi_2, \ldots \phi_n \models \psi$ to mean that any valuation that makes $\phi_1, \phi_2, \ldots \phi_n$ true also makes ψ true. $\models$ is also called the **semantic entailment** relation.

Our goal is to prove that this is equivalent to our previous notion of valid sequents.

The property that if $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$, then $\phi_1, \phi_2, \ldots \phi_n \models \psi$ is called **soundness**.

The property that if $\phi_1, \phi_2, \ldots \phi_n \models \psi$, then $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$ is called **completeness**.

Soundness states that any valid sequent says something about the relationship between the formulas involved, under all valuations.

Completeness states that given a relationship between the valuations of formulas, there must exist a proof of validity of the corresponding sequent.

It helps to think about the case where n (the number of formulas on the left-hand side) is zero. Soundness states that any formula that is a theorem is true under all valuations. Completeness says that any formula that is true under all valuations is a theorem.

We are going to prove these two properties for our system of natural deduction and our system of valuations.

The completeness and soundness of propositional logic (not the particular proof system we are studying, but an equivalent one) was shown by Emil Post in his PhD thesis in 1920. (Post went on to make contributions to early theoretical computer science, some of which you may encounter in CS 360 and CS 365.)

We will follow the proofs of completeness and soundness given in the textbook. These use the mathematical tool of induction, which was introduced in Math 135 and discussed in CS 134 and CS 136, but which deserves careful review.

Induction

Induction is used to prove some property M that depends on some integer n; that is, $M(n)$ is a statement about n. For example, $M(n)$ might be “Every even number greater than 2 but less than n is the sum of two primes”.

The principle of mathematical induction says that if $M(1)$ is true (this is the base case), and in addition we can show that for any $n \geq 1$, if $M(n)$ is true, then $M(n+1)$ is true (this is the inductive step), then we can conclude that $M(n)$ is true for all natural numbers n. (We don’t actually know how to do this for the specific statement $M(n)$ above, which is called “Goldbach’s conjecture”.)

Induction is a way of proving (in a mathematical sense) statements about an infinite number of situations or an infinite set.

We cannot prove the principle of mathematical induction; it is part of the definition of natural numbers. (We will make this idea more concrete later on in the course.)

One proof by induction you may have seen previously is the proof that the sum of the first n natural numbers is $n(n+1)/2$. This is reproduced in our textbook (Theorem 1.31).

You may have also seen that there were many different forms of induction, all equivalent to the basic form. We can vary the base case and we can vary the inductive step.

Since we will eventually be formalizing induction as a well-defined axiom of the natural numbers, we must be careful that all other notions of induction we consider follow from the basic definition, in order that our inductive reasoning will be justified.

We could, for instance, declare the base case to be proving that $M(3)$ is true, and have the inductive step show that for $n \geq 3$, $M(n)$ being true implies $M(n+1)$ being true. The result proves that $M(n)$ is true for all $n \geq 3$.

Given such a proof, we can define the property P such that $P(n) = M(n+2)$, and then mechanically translate the proof involving M to a proof involving P. The base case of that proof would be proving $P(1)$ true, and the inductive step would show that for $n \geq 3$, if $P(n-2)$ is true, then $P(n-1)$ is true, claiming a conclusion that $P(n-2)$ is true for all $n \geq 3$.

Now the additional substitution $m = n - 2$ makes this into a proof that shows that $P(1)$ is true, and for $m \geq 1$, $P(m)$ being true implies that $P(m + 1)$ is true. The conclusion becomes that $P(m)$ is true for all $m \geq 1$. This is the basic form of induction, and so this proof is valid. By reversing all our steps, we can show that the original proof is also valid.

A similar technique works to show that if we have a proof that keeps the base case of showing $M(1)$ true, but modifies the inductive step to show that for $n \geq 2$, $M(n - 1)$ being true implies $M(n)$ true. Again, a suitable set of mechanical substitutions gets us to the basic form. This form might be preferable for certain properties M where the algebra or other reasoning is simplified going from $M(n - 1)$ to $M(n)$.

There is a form called “simultaneous induction”, in which two (or more) statements are proved for all n.

As an example, we may have properties $M_1(n)$ and $M_2(n)$. The base case proves both $M_1(1)$ and $M_2(1)$. The inductive step assumes both $M_1(n)$ and $M_2(n)$, and uses these assumptions to prove both $M_1(n+1)$ and $M_2(n+1)$. The conclusion reached is that $M_1(n)$ is true for all n, and $M_2(n)$ is also true for all n.

By defining $P(n)$ to be the property “$M_1(n)$ is true and $M_2(n)$ is true”, we can mechanically reduce the above proof to the basic form.

The inductive step in basic induction is of the form "Assume $P(n)$ true and use it to prove $P(n+1)$ true." From this, we conclude that $P(n)$ is true for all n. Sometimes, the induction is phrased in the following form.

"Assume $P(n)$ is not true for all n. Let n_0 be the smallest value of n for which $P(n)$ does not hold. Thus $P(n_0)$ does not hold, but $P(n_0 - 1)$ does. If, from these two facts, we can derive a contradiction, our assumption must be wrong, and $P(n)$ must hold for all n."

This argument wraps basic induction in a proof by contradiction. It is sometimes called "proof by minimal counterexample". It shows up, for instance, in CS 341, in the discussion of greedy algorithms.

Our basic form of induction is sometimes called “weak induction”, in contrast with a form called “strong induction”. (The adjectives refer to the strength of the hypothesis in the inductive step.)

In the version of strong induction done in Math 135, the base case shows that $M(1)$ is true, and the inductive step (for $n \geq 1$) assumes $M(1), M(2), \ldots M(n)$ are true, and uses that assumption to conclude $M(n+1)$ is true. Strong induction then allows the conclusion that $M(n)$ is true for all n.

Given such a proof, we can convert it to the basic form by defining the property $P(n)$ to be “$M(m)$ is true for all $m \leq n$”. All of these forms are equivalent; we just choose the one that is most suitable for our purposes.

Finally, there is a variation called “structural induction”. You saw this in CS 134 or CS 136, where it was used to prove things about binary trees.

Recall that a binary tree is a collection of nodes that is either empty, or partitioned into three sets: a root, a left subtree, and a right subtree (the last two being binary trees).

You saw proofs of facts such as “in a non-empty binary tree, the number of leaves is one more than the number of nodes with exactly two children” with a base case of the empty tree, and an inductive step of assuming this was true of the left and right subtrees of a non-empty tree and proving it is true of the whole tree.

Structural induction, for a property of some structure defined by a recursive definition, is just strong induction on the number of times that the definition is invoked in order to justify a particular structure.

In the case of binary trees, a justification that a nonempty tree T satisfies the definition includes justifications that its left and right subtrees are also binary trees. Those justifications thus each use the definition fewer times than the justification for T.

We can do something similar to prove properties of formulas, because we have a recursive definition for them. When we use structural induction to prove a property M of a formula of the form $\phi \wedge \psi$, we assume M is true of ϕ and ψ and use that to prove M is true for $\phi \wedge \psi$. This is strong induction on the number of times the definition is used to show a formula is well-formed.

What we have called “strong induction”, the textbook calls “course-of-values induction”, and it also justifies structural induction by talking about the height of the parse tree, instead of the number of applications of the definition (which would correspond to the number of nodes in the tree).

However, these are minor details. The point is that we have structural induction to use in order to prove properties of formulas.

Structural induction will prove useful in courses such as CS 341, CS 360, and CS 466. But all of these forms of induction are just different ways of expressing basic (“weak”) induction. We choose the form that is most clear.

Soundness

Recall our definition of soundness: if $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$, then $\phi_1, \phi_2, \ldots \phi_n \models \psi$.

Soundness asserts that a valid sequent says something about valuations. A sequent is shown valid by means of a natural deduction proof. Our proof of soundness will proceed by induction on the length of this proof (the number of lines).

Note that our proofs are just flattened trees. We could have given a recursive definition of what a proof was (an application of a rule to one or more subproofs) in which case our proof by induction becomes structural induction. It might help to keep this in mind.

Thm: If $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$, then $\phi_1, \phi_2, \ldots \phi_n \models \psi$.

Pf: If a sequent has a proof, then it has a proof that uses only the core deduction rules (i.e., no MT, LEM, PBC, or $\neg\neg$i). We shall therefore assume that that the given proof of ϕ uses only core deduction rules and proceed by strong induction on the length k of a proof of the sequent $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$.

If the proof has one line, it must be of the form

1	ϕ	premise

So the sequent is $\phi \vdash \phi$, and clearly $\phi \models \phi$.

For the inductive step, we assume that soundness holds for sequents with proofs of length less than k, and prove that it holds for a sequent $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$ with a proof of length k (again, our attention is restricted to deductions involving only core rules).

The kth line of the proof must look like

k ψ justification

We have a number of cases depending on which rule is applied in this line. We'll do enough cases to get the general idea, since we have a lot of rules in natural deduction. (This proof is easier for other systems with fewer rules, but then using such systems is more difficult.)

If the kth line looks like

k ψ $\wedge$ i k_1, k_2

then ψ must be of the form $\psi_1 \wedge \psi_2$, where ψ_1 appears on line k_1, and ψ_2 appears on line k_2.

The sequent $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi_1$ has a proof of length at most k_1 (the first k_1 lines of the proof we started with), so $\phi_1, \phi_2, \ldots, \phi_n \models \psi_1$. Similarly, $\phi_1, \phi_2, \ldots, \phi_n \models \psi_2$.

What this means is that for every interpretation Φ such that $\Phi(\phi_1) = T, \ldots, \Phi(\phi_n) = T$, we also have $\Phi(\psi_1) = T$, and whenever $\Phi(\phi_1) = T, \ldots, \Phi(\phi_n) = T$, we have $\Phi(\psi_2) = T$.

So let $\Phi(\phi_1) = \cdots = \Phi(\phi_n) = T$.

Then $\Phi(\psi_1) = \Phi(\psi_2) = T$. Therefore,

$$\Phi(\psi_1 \wedge \psi_2) = \text{meaning}(\wedge)(\Phi(\psi_1), \Phi(\psi_2)) = \text{meaning}(\wedge)(T, T) = T\,.$$

Thus, $\phi_1, \phi_2, \ldots, \phi_n \models \psi$, as required.

This sort of reasoning works well for many other rules, such as $\wedge$e$_1$ and $\rightarrow$e. But it gets a little complicated if the last line in our proof closes off a box, as with $\rightarrow$i. Suppose we have such a proof of $\phi_1, \phi_2, \ldots, \phi_n \vdash \alpha \rightarrow \beta$.

1		
$\vdots$		
i	α	assumption
$\vdots$		
$k-1$	β	justification
k	$\alpha \rightarrow \beta$	$\rightarrow$ i $\quad i$–$(k-1)$

If we delete the last line, this is not a complete proof.

But we can turn it into one by making the assumption of the open box into a premise.

1		
$\vdots$		
i	α	premise
$\vdots$		
$k-1$	β	justification

This is a proof of the sequent $\phi_1, \phi_2, \ldots, \phi_n, \alpha \vdash \beta$.

Since our proof of $\phi_1, \phi_2, \ldots, \phi_n, \alpha \vdash \beta$ has $k-1$ lines, we can apply the inductive hypothesis to conclude that $\phi_1, \phi_2, \ldots, \phi_n, \alpha \models \beta$.

We need to conclude that $\phi_1, \phi_2, \ldots, \phi_n \models \alpha \rightarrow \beta$. So consider an interpretation Φ that makes $\phi_1, \phi_2, \ldots, \phi_n$ true. The only way we could have $\Phi(\alpha \rightarrow \beta) = F$ would be if $\Phi(\alpha) = T$ and $\Phi(\beta) = F$ (this is by examination of the truth table that defines $\mathrm{meaning}(\rightarrow)$). But we have

$$\phi_1, \phi_2, \ldots, \phi_n, \alpha \models \beta \ ,$$

so if $\Phi(\phi_1) = \cdots = \Phi(\phi_n) = \Phi(\alpha) = T$, then $\Phi(\beta) = T$.
Hence $\Phi(\alpha \rightarrow \beta) = T$, and we have $\phi_1, \phi_2, \ldots, \phi_n \models \alpha \rightarrow \beta$.

This sort of reasoning works for the cases where the last line closes off a proof box.

The full proof involves a complete examination of each of the dozen or so rules of natural deduction, but we have already seen the two main ideas.

Soundness gives us a method of showing that a sequent $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ does **not** have a proof in natural deduction; we simply have to find a valuation that makes $\phi_1, \phi_2, \ldots, \phi_n$ true but ψ false.

It is worth noting that the the proof of soundness given here for classical deductions actually holds for both classical and intuitionist proofs. An intuitionist proof is simply one that does not employ the ¬¬e rule; hence any intuitionist proof is also a classical proof, and therefore soundness for intuitionism is an easy consequence of soundness for classicism.

Completeness

Thm: If $\phi_1, \phi_2, \ldots \phi_n \models \psi$, then $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$.

Proof: The proof is divided into three stages.

The first stage shows that if $\phi_1, \phi_2, \ldots \phi_n \models \psi$, then $\models \phi_1 \rightarrow (\phi_2 \rightarrow (\ldots (\phi_n \rightarrow \psi) \ldots))$.

The second stage shows that if $\models \eta$, then $\vdash \eta$, for any formula η (including those looking like what the first stage produced). If $\models \eta$, then η is called a **tautology**.

The third stage shows that if $\vdash \phi_1 \rightarrow (\phi_2 \rightarrow (\ldots (\phi_n \rightarrow \psi) \ldots))$, then $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$.

The first and third stages are the easiest, so let's get them out of the way. We first need to show that if $\phi_1, \phi_2, \ldots \phi_n \models \psi$, then $\models \phi_1 \rightarrow (\phi_2 \rightarrow (\ldots (\phi_n \rightarrow \psi) \ldots))$.

How can we have $\Phi(\phi_1 \rightarrow (\phi_2 \rightarrow (\ldots (\phi_n \rightarrow \psi) \ldots))) = F$ for an interpretation Φ? We would need $\Phi(\phi_1) = T$ and $\Phi(\phi_2 \rightarrow (\phi_3 \rightarrow \ldots (\phi_n \rightarrow \psi) \ldots)) = F$. We can continue to unravel the formula, and conclude that $\Phi(\phi_1) = \Phi(\phi_2) = \ldots = \Phi(\phi_n) = T$, and $\Phi(\psi) = F$. But we know $\phi_1, \phi_2, \ldots \phi_n \models \psi$, so this is impossible, by definition of $\models$.

For the third stage, we must take a proof of $\vdash \phi_1 \rightarrow (\phi_2 \rightarrow (\ldots (\phi_n \rightarrow \psi) \ldots))$, and from it construct a proof of $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$.

This can be done simply by augmenting the proof of $\vdash \phi_1 \to (\phi_2 \to (\ldots(\phi_n \to \psi)\ldots))$ with n lines at the beginning introducing $\phi_1, \phi_2, \ldots \phi_n$ as premises, and n lines at the end, each applying $\to$e to peel off ϕ_1, then ϕ_2, and so on up to ϕ_n, which leaves ψ as the last line. The result is a proof of $\phi_1, \phi_2, \ldots \phi_n \vdash \psi$.

Technically speaking, both of these stages should be proofs by induction on n, but they are so "iterative" or algorithmic in nature that the induction will only obscure what is going on. This is also true in the second stage, but there we also must do a structural induction on formulas, so we really need to keep things as clear as possible.

The second stage is more complicated than anything we have done so far, though it is not technically difficult. Our goal is to prove that for any formula η, if $\models \eta$, then $\vdash \eta$.

Suppose η has n distinct propositional atoms. (This n is different from the one in the first and third stages.) We are going to have to construct a proof knowing only that each of the 2^n interpretations assign T to η. We do this by constructing 2^n subproofs, one for each interpretation, and then putting them all together.

An interpretation Φ assigns T or F to each of the atoms, which we'll call $p_1, p_2, \ldots, p_n$. (You can think of it as a line in a truth table for η.) Relative to Φ, we define $\hat{p}_i$ to be the formula p_i if $\Phi(p_i) = T$; otherwise $\hat{p}_i$ is the formula $\neg p_i$. Hence, for all p_i, $\Phi(\hat{p}_i) = T$.

The induction on formulas is contained in the following lemma:

Lemma (1.38): Let ϕ be a formula with propositional atoms $p_1, p_2, \ldots p_n$, and Φ an interpretation. If $\Phi(\phi) = T$, then $\hat{p}_1, \hat{p}_2, \ldots \hat{p}_n \vdash \phi$; if $\Phi(\phi) = F$, then $\hat{p}_1, \hat{p}_2, \ldots \hat{p}_n \vdash \neg\phi$.

Note: this lemma and its proof are a great example of a proof by simultaneous induction. As we shall see, we are really only interested in the statement that if $\Phi(\phi) = T$, then $\hat{p}_1, \ldots, \hat{p}_n \vdash \phi$. But in order to have enough information to prove the result inductively, we must prove both claims.

Proof of the Lemma: By structural induction on the formula ϕ.

If ϕ is an atom p, the statement of the lemma says that $p \vdash p$ and $\neg p \vdash \neg p$, which have one-line proofs.

If ϕ is of the form $\neg\phi_1$, then we may apply the inductive hypothesis to ϕ_1. If $\Phi(\phi) = T$, then $\Phi(\neg\phi_1) = T$, which means that $\Phi(\phi_1) = F$.

By the inductive hypothesis, $\hat{p}_1, \hat{p}_2, \ldots \hat{p}_n \vdash \neg\phi_1$.

But this just says $\hat{p}_1, \hat{p}_2, \ldots \hat{p}_n \vdash \phi$, as required.

If $\Phi(\phi) = F$, then $\Phi(\phi_1) = T$, and we get $\hat{p}_1, \hat{p}_2, \ldots \hat{p}_n \vdash \phi_1$. Adding an application of $\neg\neg$i gives $\hat{p}_1, \hat{p}_2, \ldots \hat{p}_n \vdash \neg\neg\phi_1$, which is $\hat{p}_1, \hat{p}_2, \ldots \hat{p}_n \vdash \neg\phi$.

In the remaining cases, ϕ is of the form $\phi_1 \circ \phi_2$. These are all similar; we will do only one here, where ϕ is of the form $\phi_1 \rightarrow \phi_2$. (Unlike with the proof of soundness, the textbook treats all the cases.)

Let $q_1, \ldots, q_k$ be the propositional atoms of ϕ_1 and $r_1, \ldots r_\ell$ be the propositional atoms of ϕ_2. First, we treat the case where $\Phi(\phi) = F$. Then $\Phi(\phi_1) = T$ and $\Phi(\phi_2) = F$.

Applying the inductive hypothesis, we have $\hat{q}_1, \ldots, \hat{q}_k \vdash \phi_1$, and $\hat{r}_1, \ldots, \hat{r}_\ell \vdash \neg\phi_2$. Since both of the sets of mini-formulas on the left-hand sides of the sequents are subsets of $\hat{p}_1 \ldots \hat{p}_n$, we know that $\hat{p}_1, \ldots, \hat{p}_n \vdash \phi_1$, and $\hat{p}_1, \ldots, \hat{p}_n \vdash \neg\phi_2$. We can put these proofs together with one application of $\wedge$i to show $\hat{p}_1, \ldots, \hat{p}_n \vdash \phi_1 \wedge \neg\phi_2$.

We've shown $\hat{p}_1, \ldots, \hat{p}_n \vdash \phi_1 \wedge \neg\phi_2$. But what we need to show is $\hat{p}_1, \ldots, \hat{p}_n \vdash \neg(\phi_1 \rightarrow \phi_2)$. All we need is a proof of $\phi_1 \wedge \neg\phi_2 \vdash \neg(\phi_1 \rightarrow \phi_2)$. The book leaves this as an exercise.

That was the subcase where $\Phi(\phi) = \Phi(\phi_1 \rightarrow \phi_2) = F$. If $\Phi(\phi) = T$, there are three possibilities for $\Phi(\phi_1)$ and $\Phi(\phi_2)$: TT, FT, and FF. Each of these, using the same sort of reasoning, generates a sequent which is close to what we need; for example, TT generates $\hat{p}_1, \ldots, \hat{p}_n \vdash \phi_1 \wedge \phi_2$, and thus generates the exercise $\phi_1 \wedge \phi_2 \vdash \phi_1 \rightarrow \phi_2$.

We then have to consider the cases where ϕ is of the form $\phi_1 \wedge \phi_2$ or $\phi_1 \vee \phi_2$. In total, eleven exercises are generated for future practice in natural deduction. That concludes the proof of the lemma, which is not difficult, but rather tedious.

Given the lemma, we can prove the main part of stage 2, which is to show that if $\models \eta$, then $\vdash \eta$. Since η is a tautology, all 2^n possible valuations make it true. Thus for every possible choice for $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n$, we can use the lemma to get a proof of $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \eta$.

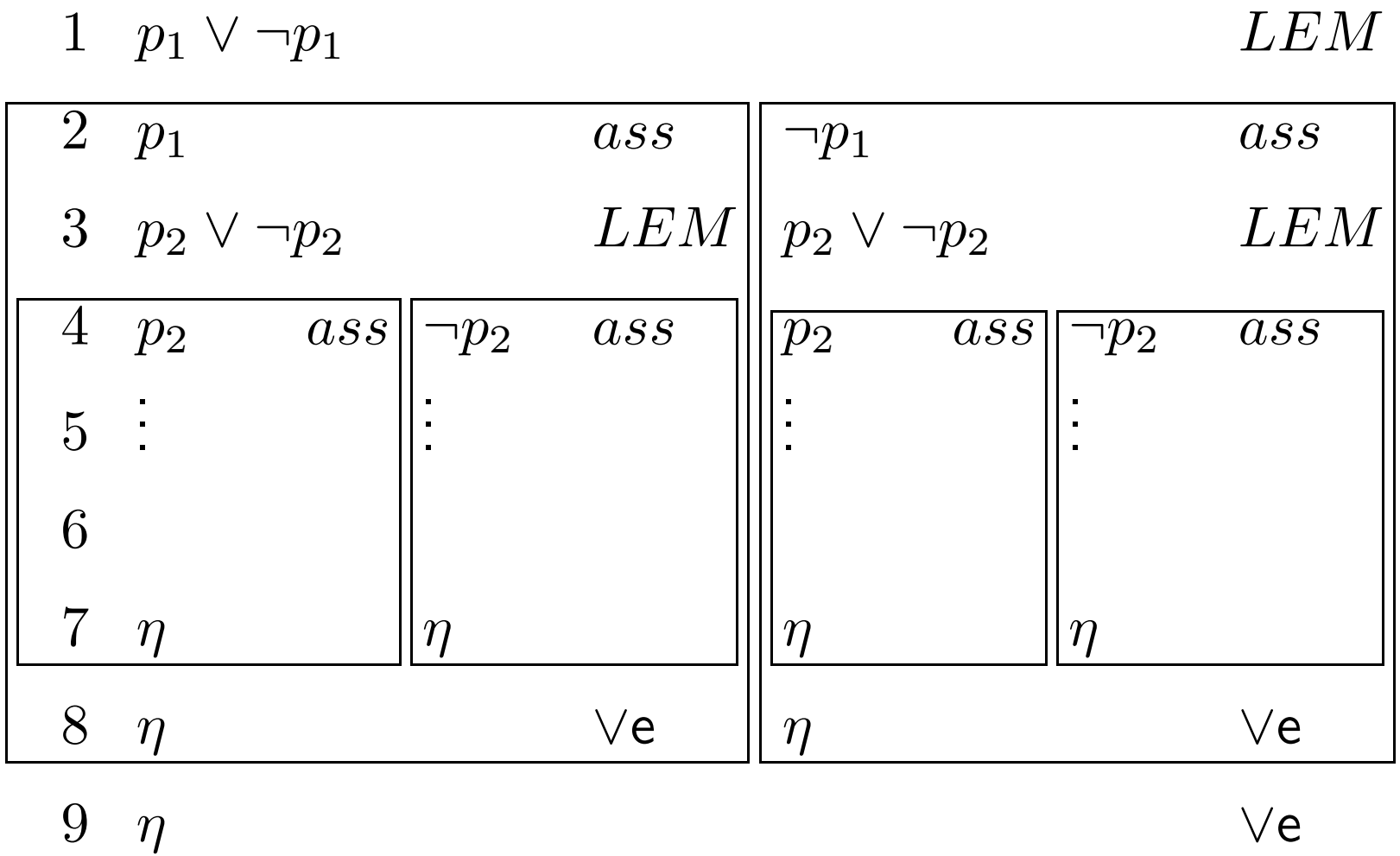
To get a proof of $\vdash \eta$, we use LEM to obtain $p_1 \vee \neg p_1$. Then we open two proof boxes with assumptions p_1 and $\neg p_1$, with the intention of using $\vee$e on them. (Though we have to do this sequentially, think of them as happening in parallel.)

Within each of those, we use LEM to obtain $p_2 \vee \neg p_2$, and do the same thing, nesting boxes. At the innermost level, we'll have 2^n boxes side by side, each one with assumptions accumulating one choice for each of $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n$.

For each one of the 2^n innermost boxes, there are n enclosing proof boxes making assumptions that build up one choice of $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n$.

Thus we can fill each one with one of the proofs provided by the lemma to conclude η within the innermost boxes. We then close the innermost proof boxes in pairs with $\vee$e, concluding η. Then we close the next matching pairs with $\vee$e, concluding η, and so on at each level, until all the proof boxes are closed.

The next slide illustrates this for the case $n = 2$.


1 $p_1 \vee \neg p_1$ LEM
2 p_1 ass
3 $p_2 \vee \neg p_2$ LEM
4 p_2 ass
5 $\vdots$
6
7 η
$\neg p_2$ ass
$\vdots$
η
8 η $\vee$e
$\neg p_1$ ass
$p_2 \vee \neg p_2$ LEM
p_2 ass
$\vdots$
η
$\neg p_2$ ass
$\vdots$
η
η $\vee$e
9 η $\vee$e

The resulting proof is not one of which we should be particularly proud. But it is a valid proof of $\vdash \eta$.

This concludes the second stage, and thus the entire proof of the completeness of propositional logic. We have shown that if $\phi_1, \phi_2, \ldots, \phi_n \models \psi$, then $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$.

Putting this together with soundness, we see that $\phi_1, \phi_2, \ldots, \phi_n \models \psi$ if and only if $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$. Our two notions, proof by natural deduction (syntactic transformations) and semantics via truth-table valuations, have been shown to coincide. The next language for which we describe a proof system and semantics, predicate logic, is also sound and complete, but the proofs of those properties are beyond the scope of this course.

What about intuitionist reasoning?

Notice how, in the proof of the completeness of propositional proof theory with respect to truth-table semantics, we made several appeals to LEM—one for each atom, in fact. Thus, the proof of completeness, as given, does not hold for intuitionist logic.

This should come as no suprise, since, upon a moment's reflection, we must realize that inuitionist logic **is not complete** with respect to truth-table semantics.

For a counterexample to completeness for intuitionist logic, it simply suffices to consider LEM. We know that LEM is not a theorem of intuitionist proof theory (i.e., $\nvdash \phi \vee \neg\phi$ under intuitionism), and yet truth-table semantics tell us that $\models \phi \vee \neg\phi$.

We could also consider as a counterexample the classical theorem $(\phi \rightarrow \psi) \vee (\psi \rightarrow \phi)$, which we discussed previously. We have already shown that this formula has a classical proof; hence, by soundness, $\models (\phi \rightarrow \psi) \vee (\psi \rightarrow \phi)$. But the statement has no intuitionist proof, so again, the formula demonstrates the incompleteness of intuitionist reasoning with respect to truth-table semantics.

It is clear, then, that any completeness proof for propositional logic with respect to truth-table semantics **must** include an appeal to LEM, or one of its equivalents ($\neg\neg e$ or PBC); otherwise, it would also be a proof of the completeness of intuitionist reasoning with respect to truth-table semantics, which we know to be false.

Intuitionist semantics?

This discussion may lead us to wonder whether we can find some other semantics for propositional logic that yields truth on exactly the intuitionist theorems.

The answer is yes, but the associated semantics is much more mathematically involved than simple truth table semantics. One way to do it is via the use of structures called **Heyting algebras** (which are a generalization of the Boolean algebra of T and F, upon which classical reasoning is based).

One specific example of a semantics for intuitionist logic is a map from each atom in a given formula into an open subset of the real plane ($\mathbb{R}^2$), rather than simply T or F. The logical connectives then have meanings phrased as operations on such sets.

Intuitionist semantics

Then a formula is an intuitionist theorem if and only if its image under this map is always $\mathbb{R}^2$, regardless of which open sets are assigned to the individual atoms.

As a result, intuitionist reasoning is both **sound** and **complete** with respect to Heyting algebra semantics.

Turning our attention once more to classical reasoning, since there are classical theorems that are not intuitionist theorems (but not vice versa), we discover that classical reasoning is **complete**, but **not sound** with respect to Heyting algebra semantics.

We can summarize the relationships between the proof systems and semantic models as follows:

	Truth-Table Semantics	Heyting Algebra Semantics
Classical Logic	Sound and Complete	Complete, but not Sound
Intuitionist Logic	Sound, but not Complete	Sound and Complete

What’s coming next?

Before we move to predicate logic, we will briefly discuss the parts of Chapter 1 that we are not covering in depth. We will also survey some alternate proof systems, and some implications of the material we have just covered, particularly as it relates to topics in other CS courses.

The definitions of both the system of natural deduction for predicate logic and the semantics associated with these more expressive formulas will prove to be considerably more complicated.

Goals of this module

You should understand the semantics presented for propositional logic in terms of interpretations, and you should be very clear on the distinction between a formula being semantically true and a formula being syntactically provable.

You should be comfortable with the various forms of induction discussed, particularly structural induction, and be able to do such proofs yourself.

You should be able to fill in any of the holes left in the proofs of soundness and completeness, both in terms of understanding the big picture (what needs to go into the holes in order to complete the proof structure), and the specific details.