1 Propositional Logic

We are going to introduce formal languages in which one can regiment mathematical and philosophical reasoning, without the distracting complexities and vagaries of natural language. In this chapter, we begin with the language of propositional logic, the logic of the connectives $\neg$, $\rightarrow$, $\wedge$, $\vee$, etc. This language is woefully inadequate for any serious applications, but it is a useful prototype to introduce general ideas and techniques that we'll also use for more powerful languages.

1.1 Syntax

When talking about language, it is important to distinguish the language that is being talked about, the *object language*, from the *meta-language* in which the talking takes place. Throughout these notes, the meta-language will be English, with added technical vocabulary that will be introduced as we go along. Our first object language is the language of propositional logic, or $\mathfrak{L}_0$.

The *primitive symbols* of $\mathfrak{L}_0$ are:

- a non-empty (and countable) set of *sentence letters*,
- the *connectives* '$\neg$' and '$\rightarrow$', and
- the parentheses, '(' and ')'.

The sentence letters are classified as *non-logical* symbols. The other expressions are *logical*. (The point of this classification will become clear in section 1.3.)

Definition 1.1

A *sentence* of $\mathfrak{L}_0$ is a finite string of symbols, built up according to the following formation rules:

(i) Every sentence letter is a sentence.
(ii) If a string A is a sentence, then so is $\neg A$.
(iii) If strings A and B are sentences, then so is $(A \rightarrow B)$.

In definition 1.1, I use 'A' and 'B' as metalinguistic variables for strings of symbols in the object language. Throughout these notes, I will often use capital letters from the beginning of the alphabet for sentences in the object language. I'll sometimes use the lowercase letters 'p' and 'q' to denote arbitrary sentence letters. I haven't said what the sentence letters of $\mathfrak{L}_0$ look like. It doesn't matter (as long as none of them has any other primitive $\mathfrak{L}_0$-symbols as a part, which I hereby stipulate). Strictly speaking, $\mathfrak{L}_0$ is therefore not a single language, but a family of languages, with different stocks of sentence letters.

Unless stated otherwise, metalinguistic variables are to be understood as universally quantified. Condition (ii) in definition 1.1, for example, doesn't talk about a particular string A, which I failed to specify. Rather, it says that *for all strings* A, if A is a sentence then $\neg A$ is a sentence. Also, by '$\neg A$', I mean the string that results by putting '$\neg$' in front of whatever string 'A' picks out. Similarly for '$(A \to B)$' and other such cases.

We introduce '$\wedge$', '$\vee$', and '$\leftrightarrow$' as metalinguistic abbreviations. That is, if A and B are $\mathfrak{L}_0$-sentences, we write

- $(A \wedge B)$ for $\neg(A \to \neg B)$;
- $(A \vee B)$ for $(\neg A \to B)$;
- $(A \leftrightarrow B)$ for $\neg((A \to B) \to \neg(B \to A))$.

We could have added '$\wedge$', '$\vee$', and '$\leftrightarrow$' as primitive symbols; you'll soon understand why we didn't. At any rate, you should remember that nothing is lost by restricting the primitive connectives to '$\neg$' and '$\to$': in classical propositional logic, all connectives can be defined in terms of these two.

It is convenient to also have a zero-ary connective $\top$ that is always true, and a dual $\bot$ that is always false. We introduce these as further metalinguistic abbreviations. Where p is an arbitrary sentence letter (say, the first in some alphabetical order), we write

- $\top$ for $p \to p$;
- $\bot$ for $\neg(p \to p)$.

Where no ambiguity threatens, I'll often omit parentheses and quotation marks. For example, I might write $A \to B$ instead of '$(A \to B)$'.

Exercise 1.1 How many sentences are there in $\mathfrak{L}_0$?

Exercise 1.2 Why did I say that no sentence letter of $\mathfrak{L}_0$ must have any other primitive $\mathfrak{L}_0$-symbol as a part? What could go wrong otherwise?

Suppose we want to show that every $\mathfrak{L}_0$-sentence has some property. The standard method for doing this is called *proof by induction on complexity*. (This sense of 'induction' is only loosely related to the kind of "inductive inference" that is often contrasted with deduction.)

Proofs by induction on complexity are based on the fact that every $\mathfrak{L}_0$-sentence is built up from sentence letters by finitely many applications of the formation rules in definition 1.1. The *complexity* of a sentence is the number of applications of these rules. Thus a sentence letter has complexity 0; $\neg p$ has complexity 1; $(p \to \neg q)$ has complexity 2; and so on. To show that every $\mathfrak{L}_0$-sentence has some property, it suffices to show two things:

(i) Every sentence of complexity 0 has the property.
(ii) *If* every sentence of complexity n has the property *then* so does every sentence of complexity $n + 1$.

Step (i) is called the *base case* of the proof; step (ii) the *inductive step*. The antecedent of (ii), that every sentence of complexity n has the property, is called the *induction hypothesis*.

As an example, let's prove that every $\mathfrak{L}_0$-sentence has an even number of parentheses.

Proposition 1.1

Every $\mathfrak{L}_0$-sentence has an even number of parentheses.

Proof by induction on complexity.

Base case. We need to show that every sentence letter has an even number of parentheses. A sentence letter has zero parentheses. Zero is even.

Inductive step. We need to show that *if* some sentences have an even number of parentheses *then* so does every sentence generated from these sentences by a single application of a formation rule from definition 1.1. We need to consider two cases, because there are two formation rules.

First, we need to show that if A has an even number of parentheses, then so does $\neg A$. This is true because $\neg A$ has the same number of parentheses as A. Second, we need to

show that if A and B have an even number of parentheses, then so does $(A \to B)$. This is true because $(A \to B)$ has two more parentheses than A and B together, and the sum of two even numbers plus two is always even. □

We'll use this method again and again, not just when we talk about sentences of $\mathfrak{L}_0$. Whenever a set of objects is generated from some base objects by finitely many applications of some operations, we can use the method to show that all of the objects have some property.

By the way: Now you can see why it is useful to have only two primitive connectives. If we had '$\wedge$', '$\vee$', and '$\leftrightarrow$' as well, we would have to check three more cases in every proof by induction on complexity.

Exercise 1.3 Prove by induction on complexity that the initial symbol of any $\mathfrak{L}_0$-sentence A never belongs to an $\mathfrak{L}_0$-sentence that is a proper part of A. (A *proper part* of a string is a substring that is not the whole string itself.)

Exercise 1.4 Prove by induction on n that for every natural number n, $0 + 1 + \ldots + n = n(n+1)/2$. That is, first show that the claim holds for $n = 0$; then show that if it holds for some n then it also holds for $n + 1$.

1.2 The propositional calculus

Next, let's explain how one may reason in $\mathfrak{L}_0$. We'll start with the ancient idea that to prove a statement means to derive it from some axioms. On this conception, a *proof system* consists of some axioms and some rules for deriving new sentences from old ones.

The first complete proof system for propositional logic, using this approach, was proposed by Gottlob Frege in 1879. I'll present a slightly simplified version, due to Jan Łukasiewicz and John von Neumann. We have three axiom schemas, meaning that every instance of these schemas is an axiom:

A1 $A \to (B \to A)$
A2 $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
A3 $(\neg A \to \neg B) \to (B \to A)$

The only rule of inference is *modus ponens*, which in this context is also known as *detachment*:

MP From A and $A \to B$ one may infer B.

I'll call this proof system *the propositional calculus*, although it is really only one of many equivalent calculi, all of which could be given that name.

Definition 1.2

A *proof* in the propositional calculus is a finite sequence of $\mathfrak{L}_0$-sentences $A_1, A_2, \ldots A_n$, each of which is either an instance of A1–A3 or follows from earlier sentences in the sequence by MP. A proof is *of* an $\mathfrak{L}_0$-sentence A if A is the last sentence in the sequence.

We write '$\vdash_0 A$' to express that there is a proof of A in the propositional calculus. Here is a proof of $p \to p$, showing that $\vdash_0 p \to p$:

1.	$p \to ((p \to p) \to p)$	Instance of A1
2.	$(p \to ((p \to p) \to p)) \to ((p \to (p \to p)) \to (p \to p))$	Instance of A2
3.	$(p \to (p \to p)) \to (p \to p)$	From 1, 2 by MP
4.	$p \to (p \to p)$	Instance of A1
5.	$p \to p$	From 3, 4 by MP

The result can be generalized. If we replace the sentence letter p by any sentence A throughout the proof, we still get a proof that conforms to definition 1.2. So we've effectively shown that $\vdash_0 A \to A$ for any sentence A.

Proof systems like our propositional calculus are called *axiomatic calculi* or *Hilbert-style calculi*. As the example illustrates, they tend to be difficult to use. They are also unnatural in that they focus on establishing logical truths. More often than not, when we turn to logic, we're interested in *consequence*: we want to know whether a certain conclusion follows from some premises, where these premises aren't logical truths. In a strict axiomatic calculus, any question about consequence must be reformulated as a question about logical truth: to test whether B is a logical consequence of $A_1, ..., A_n$, one would check whether $(A_1 \wedge ... \wedge A_n) \to B$ is a logical truth.

We can, however, also extend our calculus to handle deductions from non-logical premises.

Definition 1.3

A *deduction* of an $\mathfrak{L}_0$-sentence A from a set Γ ("gamma") of $\mathfrak{L}_0$-sentences in the propositional calculus is a finite sequence of sentences $A_1, A_2, \ldots A_n$, with $A_n = A$, each of which is either an instance of A1–A3, an element of Γ, or follows from previous sentences by MP.

We write '$\Gamma \vdash_0 A$' to express that there is a deduction of A from Γ. For example, '$\{p, q\} \vdash_0 p$' is a sentence in our metalanguage saying that p is deducible from the set containing p and q. We'll usually omit the set braces and simply write '$p, q \vdash_0 q$'.

The following *structural principles* about the $\vdash_0$ relation immediately follow from definition 1.3.

Id	$A \vdash_0 A$
Mon	If $\Gamma \vdash_0 A$ then $\Gamma, B \vdash_0 A$
Cut	If $\Gamma \vdash_0 A$ and $\Delta, A \vdash_0 B$ then $\Gamma, \Delta \vdash_0 B$

Here, 'Id' stands for 'Identity', 'Mon' for 'Monotonicity'. As usual, 'A' and 'B' range over arbitrary $\mathfrak{L}_0$-sentences; 'Γ' and 'Δ' ('delta') range over arbitrary sets of $\mathfrak{L}_0$-sentences; 'Γ, B' is shorthand for '$\Gamma \cup \{B\}$', the union of Γ and $\{B\}$. (The union of two sets is the set that contains all and only the elements that are in either of the two sets.)

Exercise 1.5 Explain why Id, Mon, and Cut follow from definition 1.3, without invoking A1–A3 or MP.

A proof (in the sense of definition 1.2) is a special case of a deduction (in the sense of definition 1.3), with an empty set of premises Γ. The following theorem shows that every deduction can be converted into a proof.

Theorem 1.1: The Deduction Theorem (DT)

If $\Gamma, A \vdash_0 B$ then $\Gamma \vdash_0 A \to B$.

Suppose there is a deduction of B from Γ. Being finite, this deduction must use only finitely many premises from Γ. Call them $A_1, \ldots, A_n$. So we have

$$A_1, \ldots, A_n \vdash_0 B.$$

By the Deduction Theorem, we can infer that

$$A_1, \ldots, A_{n-1} \vdash_0 A_n \to B.$$

Applying the theorem again, we get

$$A_1, \ldots, A_{n-2} \vdash_0 A_{n-1} \to (A_n \to B).$$

Continuing in this way, we can move all the premises to the right, until we get

$$\vdash_0 A_1 \to (A_2 \to (\ldots (A_n \to B) \ldots)).$$

$(A_1 \to (A_2 \to (\ldots (A_n \to B) \ldots))$ is provably equivalent to $(A_1 \wedge A_2 \wedge \ldots \wedge A_n) \to B$, so we also get $\vdash_0 (A_1 \wedge A_2 \wedge \ldots \wedge A_n) \to B$.)

To prove the Deduction Theorem, we use a method called *strong induction*. With strong induction, we would show that all natural numbers $0, 1, 2, 3, \ldots$ have a certain property by showing that *whenever all numbers smaller than a given number n have the property, then so does n itself.*

Why does this entail that all numbers have the property? Well, 0 must have the property: there are no natural numbers smaller than 0, so it is vacuously true that all numbers smaller than 0 have the property. Given that 0 has the property, 1 must have it as well; given that 0 and 1 have it, 2 must have it; and so on.

Proof of the Deduction Theorem.

Let $B_1, B_2, \ldots, B_n$ be a deduction of B from $\Gamma \cup \{A\}$. We shall prove by strong induction on k that $\Gamma \vdash_0 A \to B_k$ for all $k = 1, 2, \ldots, n$. Since $B_k = B$, this will establish the theorem.

We need to show that *if* $\Gamma \vdash_0 A \to B_i$ for all $i < k$, *then* $\Gamma \vdash_0 A \to B_k$. We distinguish three cases, corresponding to the ways in which B_k can appear in the deduction, according to definition 1.3.

Case 1. B_k is an axiom. By A1, $B_k \to (A \to B_k)$ is also an axiom. By MP, we can derive $A \to B_k$. So $\vdash_0 A \to B_k$, and so $\Gamma \vdash_0 A \to B_k$ by Mon.

Case 2. B_k is an element of $\Gamma \cup \{A\}$. We need to consider two subcases.

Subcase 2a. B_k is in Γ. Then $\Gamma \vdash_0 B_k$ by Id and Mon. As in case 1, we also have $\vdash_0 B_k \to (A \to B_k)$ by A1, so we get $\Gamma \vdash_0 A \to B_k$ by Mon and MP.

Subcase 2b. B_k is A. Then $A \to B_k$ is $A \to A$. We've just proved above that $\vdash_0 A \to A$. By Mon, we have $\Gamma \vdash_0 A \to A$.

Case 3. B_k follows by MP from two previous lines B_i and $B_i \to B_k$ in the deduction. By induction hypothesis, one can deduce $A \to B_i$ and $A \to (B_i \to B_k)$ from Γ. Axiom A2 gives us

$$(A \to (B_i \to B_k)) \to ((A \to B_i) \to (A \to B_k)).$$

Using A2, the deduction of $A \to B_i$ and $A \to (B_i \to B_k)$ from Γ can therefore be extended by MP to a deduction of $(A \to B_i) \to (A \to B_k)$ and from there to $A \to B_k$. □

The proof uses axioms A1 and A2. If we're interested in what can and can't be deduced in the propositional calculus, we never need to invoke A1 and A2 again: if needed, they can be recovered from DT and MP. Here is how we can recover A1 $(A \to (B \to A))$:

1.	$A \vdash_0 A$	(Id)
2.	$A, B \vdash_0 A$	(Mon)
3.	$A \vdash_0 B \to A$	(2, DT)
4.	$\vdash_0 A \to (B \to A)$	(3, DT)

Note that this is not a proof in the propositional calculus. It is a metalinguistic argument showing that a certain proof in the propositional calculus exists.

Exercise 1.6 Show in the same way that A2 can be derived from DT and MP. That is, show from the structural rules, DT, and MP that $\vdash_0 (A \to (B \to C)) \to ((A \to B) \to (A \to C))$.

Exercise 1.7 Is the converse of the DT true as well? How is it related to MP?

Now for some facts about negation.

Theorem 1.2: Ex Falso Quodlibet (EFQ)

If $\Gamma \vdash_0 A$ and $\Gamma \vdash_0 \neg A$, then $\Gamma \vdash_0 B$

Proof.

1.	$\Gamma \vdash_0 A$	(Assumption)
2.	$\Gamma \vdash_0 \neg A$	(Assumption)
3.	$\Gamma, \neg B \vdash_0 \neg A$	(2, Mon)
4.	$\Gamma \vdash_0 \neg B \to \neg A$	(3, DT)
5.	$\vdash_0 (\neg B \to \neg A) \to (A \to B)$	(A3)
6.	$\Gamma \vdash_0 A \to B$	(4, 5, MP)
7.	$\Gamma \vdash_0 B$	(1, 6, MP)

□

Theorem 1.3: Double Negation Elimination (DNE)

If $\Gamma \vdash_0 \neg\neg A$ then $\Gamma \vdash_0 A$.

Proof.

1.	$\Gamma \vdash_0 \neg\neg A$	(Assumption)
2.	$\Gamma, \neg A \vdash_0 \neg\neg A$	(1, Mon)
3.	$\Gamma, \neg A \vdash_0 \neg A$	(Id)
4.	$\Gamma, \neg A \vdash_0 A$	(2, 3, EFQ)
5.	$\Gamma \vdash_0 \neg A \to A$	(4, DT)
6.	$\Gamma, \neg A \vdash_0 \neg(\neg A \to A)$	(3, 4, EFQ)
7.	$\Gamma \vdash_0 \neg A \to \neg(\neg A \to A)$	(6, DT)
8.	$\vdash_0 (\neg A \to \neg(\neg A \to A)) \to ((\neg A \to A) \to A)$	(A3 with $B = (\neg A \to A)$)
9.	$\Gamma \vdash_0 (\neg A \to A) \to A$	(7, 8, MP)
10.	$\Gamma \vdash_0 A$	(5, 9, MP)

□

Theorem 1.4: Reductio Ad Absurdum (RAA)

If $\Gamma, A \vdash_0 B$ and $\Gamma, A \vdash_0 \neg B$, then $\Gamma \vdash_0 \neg A$.

Proof.

1.	$\Gamma, A \vdash_0 B$	(Assumption)
2.	$\Gamma, A \vdash_0 \neg B$	(Assumption)
3.	$\Gamma, A \vdash_0 \neg A$	(1, 2, EFQ)
4.	$\Gamma, \neg\neg A \vdash_0 \neg\neg A$	(Id, Mon)
5.	$\Gamma, \neg\neg A \vdash_0 A$	(4, DNE)
6.	$\Gamma, \neg\neg A \vdash_0 \neg A$	(3, 5, Cut)
7.	$\Gamma \vdash_0 \neg\neg A \to \neg A$	(6, DT)
8.	$\Gamma, \neg\neg A \vdash_0 \neg(\neg\neg A \to \neg A)$	(5, 6, EFQ)
9.	$\Gamma \vdash_0 \neg\neg A \to \neg(\neg\neg A \to \neg A)$	(8, DT)
10.	$\Gamma \vdash_0 (\neg\neg A \to \neg(\neg\neg A \to \neg A)) \to ((\neg\neg A \to \neg A) \to \neg A)$	(A3)
11.	$\Gamma \vdash_0 (\neg\neg A \to \neg A) \to \neg A$	(9, 10, MP)
12.	$\Gamma \vdash_0 \neg A$	(7, 11, MP) □

We needed A3 in the derivation of these facts. As in the case of A1 and A2, we won't need A3 any more, now that we have EFQ, DNE, and RAA. The relation $\vdash_0$ is fully characterized by the structural rules Id, Mon, Cut, together with MP, DT, EFQ, DNE, and RAA.

We could have used different axioms, or a different combination of axioms and inference rules to obtain the same result. Frege's original calculus, for example, has six axioms and an additional rule of substitution. But it is equivalent to the calculus I've introduced, since it determines the same relation $\vdash_0$,

Exercise 1.8 Show that $\Gamma \vdash_0 \bot$ iff there is a sentence A for which $\Gamma \vdash_0 A$ and $\Gamma \vdash_0 \neg A$.

Exercise 1.9 Show: (a) $\neg A \vdash_0 A \to B$. (b) $B \vdash_0 A \to B$. (c) $A \to \neg A \vdash_0 \neg A$;

Exercise 1.10 Show, by first expanding the definition of $\wedge$:

(a) If $\Gamma \vdash_0 A$ and $\Gamma \vdash_0 B$ then $\Gamma \vdash_0 A \wedge B$
(b) If $\Gamma \vdash_0 A \wedge B$ then $\Gamma \vdash_0 A$ and $\Gamma \vdash_0 B$

We can also design different types of proof systems that are equivalent to the propositional calculus. For example, you will have noticed that our metalinguistic proofs, using Id, Mon, Cut, MP, DT, etc., are generally much simpler than proofs in our official calculus. We can turn these proofs into their own calculus. Each line of a proof, in this calculus, is a *sequent* $\Gamma \vdash_0 A$. There are no axioms. Instead, we have the rules Id, Mon, Cut, MP, etc. to operate on sequents. To show that A follows from Γ, one tries to derive the sequent $\Gamma \vdash_0 A$.

I've introduced '$\Gamma \vdash_0 A$' to mean 'there is a deduction of A from Γ in the propositional calculus'. We don't want the lines in our new calculus to refer to deductions in another calculus. So we should replace '$\vdash_0$' by a different symbol. The standard choice is'$\Rightarrow$'. Also, it turns out that we can drop Mon and Cut in favour of a slightly strengthened form of Id:

Id^+ $\quad \Gamma, A \Rightarrow A$

We can also drop EFQ, as it is derivable from RAA. The remaining rules of our new calculus are:

MP	From $\Gamma \Rightarrow A$ and $\Gamma \Rightarrow A \to B$, infer $\Gamma \Rightarrow B$.
DT	From $\Gamma, A \Rightarrow B$, infer $\Gamma \Rightarrow A \to B$.
RAA	From $\Gamma, A \Rightarrow B$ and $\Gamma, A \Rightarrow \neg B$, infer $\Gamma \Rightarrow \neg A$.
DNE	From $\Gamma \Rightarrow \neg\neg A$, infer $\Gamma \Rightarrow A$.

This is a stripped-down version of the *sequent calculus* invented by Gerhard Gentzen in the 1930s. It determines the same proof relation as our propositional calculus: a sequence $\Gamma \Rightarrow A$ is provable in the sequent calculus iff there is a deduction of A from Γ in the propositional calculus.

Here is a simple schematic proof in the sequent calculus to show that $A \to B$ and $B \to C$ together entail $A \to C$.

1.	$A{\to}B,\ B{\to}C,\ A \Rightarrow A{\to}B$	Id^+
2.	$A{\to}B,\ B{\to}C,\ A \Rightarrow B{\to}C$	Id^+
3.	$A{\to}B,\ B{\to}C,\ A \Rightarrow A$	Id^+
4.	$A{\to}B,\ B{\to}C,\ A \Rightarrow B$	1, 3, MP
5.	$A{\to}B,\ B{\to}C,\ A \Rightarrow C$	2, 4, MP
6.	$A{\to}B,\ B{\to}C \Rightarrow A{\to}C$	5, DT

When writing out proofs like this, one often needs to repeat the same sentences on the left of '⇒' again and again. So-called *natural deduction* calculi introduce shortcuts to avoid these repetitions, dropping the '⇒' symbol and using lines or boxes to indicate the sentences to its left. You may have encountered such a calculus in your intro logic course. If so, you may want to write down a natural-deduction proof of the above entailment and compare it with the sequent-calculus proof. (Can you see how the two are related?) You may also have come across *tableau calculi* or *tree proof* calculi. These are, in effect, upside-down sequent proofs in which all sentences are pushed to the left of the arrow.

All these calculi are much easier to use than our propositional calculus. On the flip side, proofs in the propositional calculus are easier to *describe* than proofs in the other calculi: a proof is simply a list of $\mathfrak{L}_0$-sentences, each of which is either an instance of A1–A3 or follows from earlier sentences by MP. This makes it easier to prove metatheorems about what is or is not provable in the calculus. Since all the calculi are equivalent, and we're mostly interested in metatheorems, we'll take the propositional calculus to be the official calculus of classical propositional logic.

In this course, we'll focus on classical logic. But it is worth mentioning that there are also *non-classical* logics for $\mathfrak{L}_0$. These always drop one or more of the principles Id, Mon, Cut, MP, DT, EFQ, DNE, and RAA, and sometimes replace them by other principles. For example, *intuitionistic logic* drops DNE. This has the possibly attractive consequence that the rules for negation become self-contained in the sense that they don't allow proving any negation-free sentences that can't already be proved without them.

Exercise 1.11 Give a sequent calculus proof of *Peirce's Law*: $p \to ((p \to q) \to p)$. The proof requires DNE, although the sentence doesn't contain any negation.

1.3 Semantics

You may have noticed that I have introduced $\mathfrak{L}_0$ without saying anything about what the expressions of the language mean. Introductory logic texts often suggest that '$\neg$' and '$\to$' have roughly the same meaning as 'not' and 'if …then' in English. But we haven't built this tenuous connection to English into the formal language. In this section, we're going to study a more rigorous theory of meaning for $\mathfrak{L}_0$.

The status of this kind of theory is controversial. Some hold that the meaning of a logical expression is given by the rules for reasoning with the expression, which we've already described. This approach to meaning is sometimes called *inferential role semantics*. (*Semantics* is the study of meaning.)

The kind of theory we're going to study instead belongs to the tradition of *truth-conditional semantics*. The guiding idea of truth-conditional semantics is that the meaning of a sentence can be given by stating what (typically non-linguistic) conditions must be satisfied for the sentence to be true. The German sentence 'Schnee ist weiss', for example, is true iff snow is white, and arguably this information captures the core of its meaning. On the truth-conditional approach, the meaning of sub-sentential expressions like 'weiss' or '$\neg$' is determined by their contribution to the truth-conditions of sentences in which they occur.

If we apply this approach to $\mathfrak{L}_0$, we first need to assign truth-conditions to the sentence letters. To a first approximation, this might look as follows:

p: snow is white.
q: grass is purple.
…

Here I give the truth-conditions by using English sentences. This is not ideal, because English sentences may not have fully precise and determinate truth-conditions. (It isn't clear what, exactly, must be the case for 'snow is white' to be true.) Fortunately, we'll see in a moment that we don't need to worry about this problem because we won't really need to assign a meaning to the sentence letters after all.

Moving on, we need to explain how the logical operators contribute to the truth-conditions of sentences in which they occur. This is the important part. We do this inductively, as follows:

(i) $\neg A$ is true iff A is not true.
(ii) $A \to B$ is true iff A is not true or B is true.

To see what this is saying, let's pretend that I managed to assign precise truth-conditions to the sentence letter p. We thereby know in what kinds of scenarios p is true and in what kinds of scenarios it is false. The above statement about $\neg$ now tells us $\neg p$ is true in precisely those scenarios in which p is not true. In general, it tells us how to determine the conditions under which $\neg A$ is true based on the conditions under which A is true. Similarly, the statement about $\to$ tells us how to determine the conditions under which $A \to B$ is true based on the conditions under which A and B are true.

The truth-conditional conception of meaning is useful in logic because it ties in with a natural conception of entailment. Intuitively, some premises entail a conclusion iff the truth of the premises guarantee the truth of the conclusion; that is, there is no conceivable scenario in which the premises are true while the conclusion is false. If that's right then

knowledge of truth-conditions is exactly what we need if we want to know whether some premises entail some conclusion.

In fact, logic is about a particular type of entailment. Suppose we give the following truth-conditions to p and q:

> p: Snow is white.
> q: Snow is purple.

Then p entails $\neg q$: there is no scenario in which snow is white and also purple. The inference from p to $\neg q$ is valid, but it is not *logically* valid. That's because it depends on the meaning of the non-logical expression p and q. Logic abstracts away from the interpretation of non-logical expressions. Some premises *logically entail* a conclusion iff there's no conceivable scenario in which the premises are true and conclusion false, on any interpretation of the non-logical expressions.

Here we need a distinction between "logical" and "non-logical" expressions. This is best seen as a matter of choice. In *epistemic logic*, for example, a regimented version of 'it is known that' counts as logical. Since propositional logic is the logic of the Boolean connectives, '$\neg$' and '$\rightarrow$' here count as logical; the sentence letters are non-logical.

We now have this preliminary account of logical entailment:

> Some premises Γ logically entail a sentence A iff every scenario and interpretation of the sentence letters that makes the sentences in Γ true also makes A true.

We can render this simpler and more precise. Think of what you need to know about a pair of a scenario S and an interpretation I of the sentence letters in order to determine whether an arbitrary $\mathfrak{L}_0$-sentence – say, $\neg p$ – is true. I could tell you that p means that snow is purple, and that the scenario is one in which snow is red. You could then figure out that $\neg p$ is true (relative to S and I), using the interpretation rule for negation and the information I gave you. But you don't need all that information. It would be enough if I merely told you that p means something that isn't true in S. By the interpretation rule for negation, you could infer that $\neg p$ is true in S under I. Generalizing, all the information we need about a pair of a scenario S and an interpretation I to determine whether an arbitrary $\mathfrak{L}_0$-sentence is true in S under I is which sentence letters are true and which are false in S under I. This means that instead of quantifying over scenarios and interpretations, we can simply quantify over assignments of truth-values to the sentence letters. Such assignments are often called 'interpretations', but this jargon is misleading. We'll call them 'models'.

Definition 1.4: Model

A *model* for $\mathfrak{L}_0$ is an assignment σ ("sigma") of truth-values to the sentence letters of $\mathfrak{L}_0$.

That is, a model is a function σ that assigns to each sentence letter p one of the two truth values, which I'll label 'T' and 'F'. We use standard function notation here, using '$\sigma(p) = T$' to express that σ assigns the value T to p and '$\sigma(p) = F$' to express that σ assigns F to p.

Next, we explain how an assignment of truth-values to sentence letters determines an assignment of truth-values to all sentences of $\mathfrak{L}_0$, in accordance with our above interpretation rules for '$\neg$' and '$\rightarrow$'. We write '$\sigma \Vdash A$' (read: "σ satisfies A") to mean that A is true in the model σ, and '$\sigma \nVdash A$' to mean that A is not true in σ. The satisfaction relation $\Vdash$ is defined as follows:

Definition 1.5

Let σ be a model for $\mathfrak{L}_0$. For any sentence letter p and sentences A and B:

(i) $\sigma \Vdash p$ iff $\sigma(p) = T$.
(ii) $\sigma \Vdash \neg A$ iff $\sigma \nVdash A$.
(iii) $\sigma \Vdash A \rightarrow B$ iff $\sigma \nVdash A$ or $\sigma \Vdash B$.

For example, if $\sigma(p) = T$ and $\sigma(q) = F$, then $\sigma \Vdash p \rightarrow (q \rightarrow \neg q)$, as you can confirm by working through definition 1.5.

We can now define logical entailment, as already announced:

Definition 1.6

Sentences Γ *(logically) entail* a sentence A (for short, $\Gamma \models A$) iff every model that satisfies every sentence in Γ also satisfies A.

We allow Γ to be infinite, and to be empty. If something is (logically) entailed by the empty set of premises, it is called *(logically) valid*. Since our topic is logic, I'll drop 'logically' when talking about validity and entailment from now on.

Definition 1.7
A is *valid* (for short, $\models A$) iff every model satisfies A.

Exercise 1.12 Explain why, if $\Gamma \models A$ by definition 1.6, then Γ entails A according to the earlier, informal definition.

Exercise 1.13 Show that A is valid iff A is entailed by $\emptyset$.

Exercise 1.14 Show that all instances of A1–A3 are valid.

Exercise 1.15 We have now introduced five arrow-like symbols: $\rightarrow$, $\vdash_0$, $\Rightarrow$, $\Vdash$, and $\models$. Explain what each of them means and to which language it belongs. (For the record: we will never use $\Rightarrow$ again.)

1.4 Soundness and Completeness

We have explored two perspectives on logic. The first was *proof-theoretic*. We studied proofs and deductions as arrangements of symbols conforming to certain rules, without any extrinsic concern for what the symbols might mean. We then turned to a *model-theoretic* perspective, defining notions of validity and entailment in purely semantic terms.

Ideally, we'd like the two perspectives to harmonize: a sentence should be provable iff it is valid. More generally, we should have $\Gamma \vdash_0 A$ iff $\Gamma \models A$. In this section, we will show that this is indeed the case.

We have two directions to check. We first show that if $\Gamma \vdash_0 A$ then $\Gamma \models A$. This shows that the propositional calculus, and all the calculi equivalent to it, are *sound* with respect to the model-theoretic conception of entailment: anything that can be deduced from some premises in the calculus is entailed by the premises.

Afterwards, we'll show the converse, that if $\Gamma \models A$ then $\Gamma \vdash_0 A$. This shows that the calculus is *complete*: whenever something is entailed by some premises, it can be deduced from the premises.

The soundness proof is straightforward.

Theorem 1.5: Soundness of the propositional calculus

If $\Gamma \vdash_0 A$, then $\Gamma \models A$.

Suppose $\Gamma \vdash_0 A$. This means that there is a sequence $A_1, A_2, \ldots, A_n$ such that $A_n = A$ and each A_k in the sequence is either an axiom, a member of Γ, or follows from previous sentences by MP. We show by strong induction on k that $\Gamma \models A_k$. The theorem then follows by taking $k = n$.

Case 1. A_k is an axiom. Then $\Gamma \models A_k$ by exercise 1.14.

Case 2. A_k is a member of Γ. Then $\Gamma \models A_k$ holds trivially.

Case 3. A_k follows from previous sentences A_i and $A_i \to A_k$ by MP. By induction hypothesis, $\Gamma \models A_i$ and $\Gamma \models A_i \to A_k$. It follows by clause (iii) of definition 1.5 that $\Gamma \models A_k$. □

Completeness is harder. The first completeness proof for a propositional calculus was given by Paul Bernays in 1918. We're going to use a different technique, due to Leon Henkin (1949), that works for a wide range of logics. We'll use it again in chapter 3 to prove completeness for first-order logic.

Before we start, we need to define two key concepts. Let Γ be a set of $\mathfrak{L}_0$-sentences. We'll say that Γ is *consistent* if one can't deduce a contradiction from it: there is no sentence A such that $\Gamma \vdash_0 A$ and $\Gamma \vdash_0 \neg A$; equivalently, by exercise 1.8: $\Gamma \nvdash_0 \bot$. We say that Γ is *satisfiable* if there is some model that satisfies every sentence in Γ.

The following lemmas allow us to reformulate completeness in terms of consistency and satisfiability.

Lemma 1.1

$\Gamma \cup \{\neg A\}$ is satisfiable iff $\Gamma \not\models A$.

Proof. Immediate from definitions 1.5 and 1.6. □

Lemma 1.2

$\Gamma \nvdash_0 A$ iff $\Gamma \cup \{\neg A\}$ is consistent.

Proof. Suppose $\Gamma \cup \{\neg A\}$ is inconsistent. Then $\Gamma \vdash_0 \neg\neg A$ by RAA and so $\Gamma \vdash_0 A$ by DNE. Contraposing, this means that if $\Gamma \nvdash_0 A$ then $\Gamma \cup \{\neg A\}$ is consistent. Conversely, suppose $\Gamma \vdash_0 A$. Then $\Gamma, \neg A \vdash_0 A$ by Mon and $\Gamma, \neg A \vdash_0 \neg A$ by Id. So $\Gamma \cup \{\neg A\}$ is inconsistent. □

Now, completeness requires that whenever $\Gamma \models A$ then $\Gamma \vdash_0 A$. Equivalently, by contraposition: Whenever $\Gamma \nvdash_0 A$ then $\Gamma \not\models A$. By lemma 1.2, $\Gamma \nvdash_0 A$ iff $\Gamma \cup \{\neg A\}$ is consistent. By lemma 1.1, $\Gamma \not\models A$ iff $\Gamma \cup \{\neg A\}$ is satisfiable. So what remains to be shown to establish completeness is this: *Every consistent set of sentences is satisfiable.*

We are going to prove this in two steps. First, we show that every consistent set can be extended to a maximal consistent set. A set is *maximal consistent* if it is consistent and contains either A or $\neg A$, for each $\mathfrak{L}_0$-sentence A. Then we show that every maximal consistent set is satisfied by a model that makes true all and only the sentence letters in the set.

En route to the first step, we start with an easy observation.

Lemma 1.3

If Γ is consistent, then for any sentence A, either $\Gamma \cup \{A\}$ or $\Gamma \cup \{\neg A\}$ is consistent.

Proof. Suppose for reductio that $\Gamma \cup \{A\}$ is inconsistent, and so is $\Gamma \cup \{\neg A\}$. By RAA, it follows from the first assumption that $\Gamma \vdash_0 \neg A$, and from the second that $\Gamma \vdash_0 \neg\neg A$. So Γ is inconsistent. □

Now the first step:

Lemma 1.4: Lindenbaum's Lemma

Every consistent set is a subset of some maximal consistent set.

Let Γ_0 be some consistent set of sentences. Let $S_1, S_2, \ldots$ be a list of all $\mathfrak{L}_0$-sentences (in some arbitrary order). For every number $i \geq 0$, define

$$\Gamma_{i+1} = \begin{cases} \Gamma_i \cup \{S_i\} & \text{if } \Gamma_i \cup \{S_i\} \text{ is consistent} \\ \Gamma_i \cup \{\neg S_i\} & \text{otherwise.} \end{cases}$$

This gives us an infinite list of sets $\Gamma_0, \Gamma_1, \Gamma_2, \ldots$. We show by induction that each set in the list is consistent.

Base case. Γ_0 is consistent by assumption.

Inductive step. We assume that some set Γ_i in the list is consistent, and show that Γ_{i+1} is consistent. By lemma 1.3, either $\Gamma_i \cup \{S_i\}$ or $\Gamma_i \cup \{\neg S_i\}$ is consistent. If $\Gamma_i \cup \{S_i\}$ is consistent, then Γ_{i+1} is $\Gamma_i \cup \{S_i\}$ (by construction), so Γ_{i+1} is consistent. If $\Gamma_i \cup \{S_i\}$ is not consistent, then Γ_{i+1} is $\Gamma_i \cup \{\neg S_i\}$, so again Γ_{i+1} is consistent.

So all of $\Gamma_0, \Gamma_1, \Gamma_2, \ldots$ are consistent. Now let Γ be the set of sentences that occur in at least one of the sets $\Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3 \ldots$. (That is, let Γ be the union of $\Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3, \ldots$.) Evidently, Γ is maximal and Γ_0 is a subset of Γ. It remains to show that Γ is consistent.

Suppose not (for reductio). Then there are sentences $A_1, \ldots, A_n$ in Γ from which $\bot$ is deducible. All of these sentences have to occur somewhere on the list $S_1, S_2, \ldots$. Let S_j be the first sentence from $S_1, S_2, \ldots$ that occurs after all the $A_1, \ldots, A_n$. Since all $A_1, \ldots, A_n$ are in Γ, they have to be in Γ_j. So Γ_j is inconsistent. But we've seen that all of $\Gamma_0, \Gamma_1, \Gamma_2, \ldots$ are consistent. □

For the second step, we also need a preliminary observation:

Lemma 1.5

If Γ is maximal consistent and $\Gamma \vdash_0 A$, then $A \in \Gamma$.

Proof. If $A \notin \Gamma$, then $\neg A \in \Gamma$ by maximality. We then have $\Gamma \vdash_0 A$ and $\Gamma \vdash_0 \neg A$, contradicting consistency. □

Here comes step 2.

Lemma 1.6: Truth Lemma

Every maximal consistent set Γ is satisfied by the model σ_Γ that assigns T to every sentence letter in Γ and F to every other sentence letter.

Proof. We show that for every $\mathfrak{L}_0$-sentence A, $\sigma_\Gamma \Vdash A$ iff $A \in \Gamma$. The proof is by induction on the complexity of A.

Base case: A is a sentence letter. Then the claim directly follows from the construction of σ_Γ.

Inductive step. We consider the two cases for complex sentences. Assume first that A is $\neg B$, for some sentence B. We have to show that $\sigma_\Gamma \Vdash \neg B$ iff $\neg B \in \Gamma$. Left to

right: Assume $\sigma_\Gamma \Vdash \neg B$. Then $\sigma_\Gamma \nVdash B$ by definition 1.5. By induction hypothesis, it follows that $B \notin \Gamma$. Then $\neg B \in \Gamma$ because Γ is maximal. Right to left: Assume $\neg B \in \Gamma$. Then $B \notin \Gamma$ because Γ is consistent. By induction hypothesis, it follows that $\sigma_\Gamma \nVdash B$. So $\sigma_\Gamma \Vdash \neg B$ by definition 1.5.

Now assume that A is $B \to C$, for some sentences B and C. We show that $\sigma_\Gamma \Vdash B \to C$ iff $B \to C \in \Gamma$. Left to right: Assume $\sigma_\Gamma \Vdash B \to C$. Then $\sigma_\Gamma \nVdash B$ or $\sigma_\Gamma \Vdash C$ by definition 1.5. If $\sigma_\Gamma \nVdash B$ then $B \notin \Gamma$ by induction hypothesis; so $\neg B \in \Gamma$ by maximality and so $B \to C \in \Gamma$ by lemma 1.5 and exercise 1.9(a). If $\sigma_\Gamma \Vdash C$ then $C \in \Gamma$ by induction hypothesis, and so $B \to C \in \Gamma$ by lemma 1.5 and exercise 1.9(b). Right to left: Assume $B \to C \in \Gamma$. Assume first that B is also in Γ. Then $C \in \Gamma$ by lemma 1.5 and MP. By induction hypothesis, $\sigma_\Gamma \Vdash C$, and so $\sigma_\Gamma \Vdash B \to C$ by definition 1.5. Assume, alternatively, that B is not in Γ. By induction hypothesis, then $\sigma_\Gamma \nVdash B$, and again $\sigma_\Gamma \Vdash B \to C$ by definition 1.5. □

Let's put the pieces together:

Theorem 1.6: Completeness of the propositional calculus

If $\Gamma \models A$ then $\Gamma \vdash_0 A$.

Proof by contraposition. Assume $\Gamma \nvdash_0 A$. Then $\Gamma \cup \{\neg A\}$ is consistent by lemma 1.2. By lemma 1.4, $\Gamma \cup \{\neg A\}$ is contained in a maximal consistent set Γ^+. By lemma 1.6, there is a model σ_{Γ^+} that satisfies Γ^+ and hence $\Gamma \cup \{\neg A\}$. So $\Gamma \not\models A$. □

Exercise 1.16 Suppose we have two proof systems $\vdash_1$ and $\vdash_2$ such that whenever $\Gamma \vdash_1 A$ then $\Gamma \vdash_2 A$. Does the soundness of one system imply the soundness of the other? If so, in which direction? How about completeness?

Exercise 1.17 Someone might worry that the propositional calculus is inconsistent in the sense that it allows proving $\bot$ (from no premises). Can you allay this worry?

Exercise 1.18 Show that if we add any further axiom schema to A1–A3 that is not already provable in the propositional calculus, then we get an inconsistent

calculus. (This means that the calculus is *Post-complete*, after Emil Post, who first proved the present fact in 1921.)
Hint: By the completeness theorem, any schema that isn't provable in the calculus has invalid instances. Can you see why a schema with invalid instances must have inconsistent instances?