Introduction to Formal Logic

Masahiro Yamada

About This Book

This is material I have used to teach introductory level formal logic at Pomona College. The material is broken down to be doable in a 15 week semester with two sessions per week.

The proof system used in the book is the one Gentzen presents in his (1936).

You can get exercises and some handouts at:

`https://adamay909.github.io/logicbook/`

There is a proof checker available at:

`https://adamay909.github.io/logicTools/`

The LaTeX sources are available at:

`https://github.com/adamay909/logicbook`

License

Contents

Part I

Formal Logic

Chapter 1

Logical Connectives in English

1.1 Sentences

In this chapter we will be looking at logical connectives in a natural language, in particular English. Just exactly what logical connectives are will be explained as we proceed.

1.1.1 Sentences

Our natural languages have components that can be combined to form larger linguistic units. For example, we can combine words to form sentences. Sentences to form paragraphs. In this part of our class, our interest is in a certain type of sentences. Which type? In English there is an easy way of isolating the type of sentence we are interested in: prefixing them with 'It is true that' results in a grammatically correct sentence. Let me explain:

In English, the following are all sentences:

- Grass is always greener on the other side.
- Is your neighbor's lawn greener than yours?
- Let's go check out our neighbor's lawn.

If you prefix each of these with 'It is true that', you get:

- It is true that grass is always greener on the other side.
- It is true that is your neighbor's lawn greener than yours?
- It is true that let's go check out our neighbor's lawn.

Of the latter group, only the first is a grammatically correct sentence of English so the only of interest to us in the former group is the first sentence. It is the only one of the three that grammatically speaking could be true or false in the sense that English grammar allows us to say that it is true by simply prefixing 'it is true that' to it. From now on, when we speak of sentences, we only mean such sentences. The following are all sentences:

- One plus one is two.

- Vanilla ice cream is delicious.
- I don't like steak to be well done.
- The Earth is flat.
- If the Mongol Empire had not expanded into Eastern Europe, Russia as we know it would not have existed.
- Free will is an illusion.
- The Middle Ages ended with the fall of Constantinople.
- It is morally impermissible to tell a lie.

Each of these results in a grammatically correct sentence of English when prefixed with 'it is true that':

- It is true that one plus one is two.
- It is true that vanilla ice cream is delicious.
- It is true that I don't like steak to be well done.
- It is true that the Earth is flat.
- It is true that if the Mongol Empire had not expanded into Eastern Europe, Russia as we know it would not have existed.
- It is true that free will is an illusion.
- It is true that the Middle Ages ended with the fall of Constantinople.
- It is true that it is morally impermissible to tell a lie.

That we can prefix a sentence with 'it is true that' and get a grammatically correct sentence means just that: the resulting sentence is grammatically correct. It does not mean that the sentence is true. For instance, in the list just given, some are true, others definitely false, some controversial, and in some cases it might even be unclear whether what's being asserted really is the sort of thing that can be true or false. But that does not matter for our purposes.[1]

[1]English has a rather simple grammar that allows just prefixing a sentence with 'it is true that' to yield a grammatically correct sentence. Your native tongue might not allow anything this simple to isolate the sentences we are interested in. When in doubt about which of the sentences in your native tongue are of interest, the simple solution is to translate them into English and see.

1.1.2 Connectives

The prefix 'it is true that' is a device that we can use to form a new sentence out of another sentence. There are other such devices. Some of them can take two sentences and form a new sentence out of them. We call devices that can be used to form new sentences out of other sentences *connectives*. For example, 'it is true that' is a connective. There are also others.

For example, take the following two sentences:

1. Henry VIII was surrounded by men named Thomas.[2]
2. Henry VIII was too lazy to remember names.

These can be combined in various ways to form more complex sentences:

3. Henry VIII was surrounded by men named Thomas *and* Henry VIII was too lazy to remember names.
4. Henry VIII was surrounded by men named Thomas *because* Henry VIII was too lazy to remember names.

We can also take a single sentence and turn it into another sentence by, for instance, adding a prefix:

5. *It is not true that* Henry VIII was surrounded by men named Thomas.
6. *Anne believes that* Henry VIII was surrounded by men named Thomas.

Notice that the results are all sentences in our sense since they can all be prefixed with 'it is true that'. For example:

7. It is true that Henry VIII was surrounded by men named Thomas *and* Henry VIII was too lazy to remember names.
8. It is true that Henry VIII was surrounded by men named Thomas *because* Henry VIII was too lazy to remember names.

[2]Here are some of the people who were at one point or another of great influence under Henry VIII: Thomas Wolsey, Thomas More, Thomas Cromwell, Thomas Boleyn, Thomas Howard, Thomas Seymour, Thomas Cranmer, Thomas Wriothesley.

9. It is true that *it is not true that* Henry VIII was surrounded by men named Thomas.

10. It is true that *Anne believes that* Henry VIII was surrounded by men named Thomas.

So 'and', 'because', 'it is not true that', and 'Anne believes that' are all connectives. There are many more connectives in English. Sentences that are formed from other sentences through the use of connectives are called *compound* sentences. Sentences 3 through 10 are all compound sentences.

We will only be interested in some connectives. Of the ones we have seen so far, we will be interested in 'and' and 'it is not the case that' because these qualify as logical connectives. But we will not be interested in 'because' and 'Anne believes that' because these do not qualify as logical connectives. Just why that is so will become clearer as we move along.

Here is some terminology that we will be using:

- We will be speaking of *atomic* sentences and *compound* sentences. Atomic sentences are the sentences that we use as basic building blocks for forming compound sentences. Compound sentences are formed by manipulating atomic sentences using connectives (like combining them with the word 'and', or prefixing a sentence with 'it is not the case that').

- We will speak of *sentences* to cover both atomic and compound sentences.

- If a sentence is true, we will say that its *truth value* is true. If the sentence is not true, we will say that its truth value is false.

Just like atoms that chemistry talks about are not at all simple, atomic sentences need not be simple sentences. But when we designate a sentence as atomic, we will disregard any internal structures that the atomic sentence might have—in particular, we disregard whether they might themselves have parts that are sentences. So calling a sentence 'atomic' does not say anything about the sentence except that we are treating it as a basic building block out of which we form other sentences. What the sentence says might be highly complex.

1.2 Necessary and Sufficient Conditions

We will often be talking about necessary and sufficient conditions for this and that to be the case.

A *necessary condition* is a condition that must be met for something to be the case. For instance, in order to vote in congressional elections in the U.S. you must be 18 years or older. So being 18 years or older is a necessary condition for it to be the case that you are eligible to vote in congressional elections. The standard way to express a necessary condition in this course will be:

_____only if _____

The blanks need to be filled in so that the result is a grammatically correct sentence of English. For instance:

You are eligible to vote in U.S. congressional elections **only if** you are 18 years or older.

The bold face is just so you see the 'only if'. It will usually not be bolded or otherwise emphasized.

A *sufficient condition* for something to be the case is a condition that suffices to make it the case but is not necessary. For instance, suppose that satisfactorily passing a third semester French course suffices to meet your school's foreign language requirements for graduation but that it is not necessary to pass a third semester French course. You could take Chinese, Arabic, etc. Maybe there are even ways of avoiding a language course. The standard way of expressing a sufficient condition in this course will be:

_____if _____

For instance:

You meet the foreign language requirement **if** you satisfactorily pass a third semester French course.

Some conditions are both necessary and sufficient. The standard way of expressing that is:

_____if, and only if, _____

For instance,

You meet the mathematical and formal reasoning requirement at your school **if**, **and only if**, you take a course in mathematics, computer science, formal logic, or statistics.

'if, and only if,' is often abbreviated as 'iff.', like this:

You meet the mathematical and formal reasoning requirement at your school **iff**. you take a course in mathematics, computer science, formal logic, or statistics.

1.3 Simple Symbolization

Let us use the following sentences as our atomic sentences:

1. Snow is white.
2. The moon is made of cheese.
3. Marco Polo visited China.
4. Marco Polo met people who had visited China.
5. Marco Polo completely made up his story.

Until further notice, these are the only atomic sentences there are.

The following are all compound sentences that can be formed from these atomic sentences. I am putting brackets around the sentences making up the compound sentence. The brackets can also be nested when compound sentences are combined to form further compound sentences.

6. ⟨Snow is white⟩ and ⟨the moon is made of cheese⟩.
7. It is not the case that ⟨the moon is made of cheese⟩.
8. Either ⟨Marco Polo visited China⟩ or ⟨Marco Polo met people who had visited China⟩.

Spelling out the sentences can get tedious quickly so let's use upper case Roman letters (A, B, C, ...) to represent sentences. For example,

A: Snow is white.
B: The moon is made of cheese.
C: Marco Polo visited China.
D: Marco Polo met people who visited China.
E: Marco Polo completely made up his story.

With these symbolization keys, we can rewrite the compound sentences above as:

9. A and B.
10. It is not the case that B.
11. Either C or D.

We can also produce more complex sentences:

12. ⟨A and ⟨C or D⟩⟩ or ⟨B and ⟨⟨it is not the case that C⟩ and E⟩⟩ .

The brackets are used to make clearer what goes with what. Notice that if we had stipulated that only A through D are atomic sentences, the last sentence would not be one of the compound sentences that can be formed: it needs E and that would not be available as an atomic sentence. So changing which sentences are the atomic sentences makes a difference to what compound sentences can be formed. That is why it is important to be explicit about what the atomic sentences are.

1.4 Logical Connectives

Let's take a closer look at the connective 'and'. It takes two sentences and produces a third one. Let's use the following symbolization keys:

A: Henry VIII was surrounded by men named Thomas.
B: Henry VIII was too lazy to remember names.

A and B are both sentences. That means either can be true or false. They could be both true, both false, or one of them true and the other false. That gives us four possibilities. For each possibility, we can say whether ⟨A and B⟩ is true or not:

- A and B both true: ⟨A and B⟩ is true.
- A false and B true: ⟨A and B⟩ is false.
- A true and B false: ⟨A and B⟩ is false.
- A and B both false: ⟨A and B⟩ is false.

This means the truth value of ⟨A and B⟩ is determined by the truth values of A and B. This is an important feature of 'and'. Not all connectives are like this. For instance, consider the connective 'because'. Even if both A and B are true, that does not determine whether ⟨A because B⟩ is true: even if both A and B are true, that does not mean that the reason Henry was surrounded by Thomases is that he was too lazy to remember names—maybe Thomas happened to be an extremely popular name at the time.

When the truth value of a compound sentence using a connective is determined by the truth values of its component sentences, the connective is said to be truth-functional.

For the remainder of this course, a *logical connective* is a truth-functional connective.[3] This means that 'and' is a logical connective, 'because' is not. Moreover, when we speak of *compound sentences*, we will mean compound sentences constructed using only logical connectives unless otherwise noted.

We can represent how the connective 'and' works by the following table:

[3] In more advanced areas of logic, there are logical connectives that are not truth-functional but we will not be discussing them.

A	B	⟨A and B⟩
T	*T*	*T*
F	*T*	*F*
T	*F*	*F*
F	*F*	*F*

I am putting the compound sentence inside brackets like ⟨A and B⟩ to make clear that I am talking about the whole compound sentence. Each row of the table shows a possible combination of the truth values of A and B and the truth value of ⟨A and B⟩ for that combination of truth values for the atomic sentences. Let's call each possible combination of the truth values for the atomic sentences an *interpretation*. Note that an interpretation determines the truth value of each and every atomic sentence. Let us say that the table gives us the *truth condition* of the compound sentence ⟨A and B⟩.

The difference between 'and' and 'because' is that you cannot produce a complete table like this for the latter. If you tried, you would get:

A	B	⟨A because B⟩
T	*T*	*??*
F	*T*	*F*
T	*F*	*F*
F	*F*	*F*

It is plausible that if at least one of A and B is false, ⟨A because B⟩ is also false. But if both A and B are true, that does not determine whether ⟨A because B⟩ is true. A logical connective must be such that what goes into each cell of the truth table is determinate.

Are there logical connectives beside 'and'? There are. For instance, 'it is not the case that' is a logical connective. The truth condition for 'it is not the case that A' is given by:

A	B	⟨it is not the case that A⟩
T	*T*	*F*
F	*T*	*T*
T	*F*	*F*
F	*F*	*T*

Notice that ⟨it is not the case that A⟩ is true if and only if A is false (second

and fourth interpretation). The truth value of B does not matter. So we can use a smaller table:

A	⟨it is not the case that A⟩
T	*F*
F	*T*

Another logical connective is 'or'. The truth condition for ⟨A or B⟩ is given by:

A	B	⟨A or B⟩
T	*T*	*T*
F	*T*	*T*
T	*F*	*T*
F	*F*	*F*

⟨A or B⟩ is true if at least one of A and B is true.

We will be making heavy use of tables like this. In order to facilitate checking the correctness of tables and comparing results, let us stick to a particular way of listing the interpretations. For tables with two atomic sentences, the interpretations will always be listed as above. We will see a more general procedure for listing all the interpretations for arbitrary number of atomic sentences in the next chapter. For now, get used to the above order of listing interpretations in the case of two atomic sentences.

What about more complex compound sentences? Let's take a look at ⟨it is not the case that ⟨A and B⟩⟩. Can we state its truth condition? We can:

A	B	⟨A and B⟩	it is not the case that ⟨A and B⟩
T	*T*	*T*	*F*
F	*T*	*F*	*T*
T	*F*	*F*	*T*
F	*F*	*F*	*T*

The third column tells us the truth value of ⟨A and B⟩ for each interpretation, and the last column the truth value of ⟨it is not the case that ⟨A and B⟩⟩ for each interpretation. As you would expect, it is false iff. both A and B are true, and that happens only on the first interpretation.

Because of the way logical connectives work, any compound sentence constructed using only logical connectives is going to be such that its truth value is determined by the truth values of the atomic sentences involved. You see an example of that in the above example of ⟨it is not the case that ⟨A and B⟩⟩ . And this has important implications which we will discuss in the next section.

1.5 All the Possible Truth Conditions

⟨A and B⟩ and ⟨A or B⟩ have different truth conditions. And because they have different truth conditions, when you assert ⟨A and B⟩ you assert something different than when you assert ⟨A or B⟩. What do you assert when you assert ⟨A and B⟩? Well, you assert the world is such that both A and B are true. Of course, you might be mistaken about that, but it is what you assert. When you assert ⟨A or B⟩, you assert that at least one of A and B is true but you are not going as far as asserting that both are true—you are not excluding that possibility but you are not committed to that—and that is why asserting ⟨A or B⟩ is different from asserting ⟨A and B⟩.

Thus, when you assert one sentence and then another and you actually are asserting two things, the two sentences must have different truth-conditions. For in asserting a sentence, you are asserting that the world is a certain way, viz., one of the ways in which the sentence can be true. If you assert a second sentence, you are asserting that the world is such that one of the ways in which the second sentence can be true obtains. If the two sentences have the same truth condition, then in asserting the two sentences, you are not asserting anything different about the world: the way the world is according to your first assertion is identical to the way the world is according to your second assertion. You are saying the same thing twice over, in perhaps different ways. Two sentences with the same truth condition 'say the same thing.'

Here is a question: given two atomic sentences, how many different things could you say using compound sentences?

Consider an arbitrary compound sentence and its truth condition. As noted earlier, the truth value of the compound sentence is determined by the truth values of the atomic sentences. That is, for each interpretation the compound sentence is either true or false. And that means given two atomic sentences, the following are the possible ways the truth-condition of the compound sentence could be:

- The sentence is false in any of the four interpretations.
- The sentence is true in only one of the four interpretation.
- The sentence is true in two of the four interpretations.
- The sentence is true in three of the four interpretations.

- The sentence is true in any of the four interpretations.

Here are some examples. Let us write ⟨A⊗B⟩ to stand for some compound sentence. ⟨A⊗B⟩ could stand for a very simple compound sentence like ⟨A or B⟩ but could be any compound sentence that can be formed using logical connectives. Also, it could stand for a compound sentence that does not use both atomic sentences as components like ⟨it is not the case that A⟩. That is to say ⟨A⊗B⟩ is simply any compound sentence that can be formed given the availability of A and B as atomic sentences.

Suppose the sentence ⟨A⊗B⟩ is true in exactly one of the possible interpretations. In that case, its truth condition must be given by one of the following tables:

A	B	⟨A⊗B⟩
T	*T*	*T*
F	*T*	*F*
T	*F*	*F*
F	*F*	*F*

A	B	⟨A⊗B⟩
T	*T*	*F*
F	*T*	*T*
T	*F*	*F*
F	*F*	*F*

A	B	⟨A⊗B⟩
T	*T*	*F*
F	*T*	*F*
T	*F*	*T*
F	*F*	*F*

A	B	⟨A⊗B⟩
T	*T*	*F*
F	*T*	*F*
T	*F*	*F*
F	*F*	*T*

Here is an example of a truth condition that make a sentence come out true in two possible interpretations:

A	B	⟨A⊗B⟩
T	*T*	*T*
F	*T*	*F*
T	*F*	*F*
F	*F*	*T*

Given this table, the sentence is true if A and B are both true or both false. There are altogether six ways in which a compound sentence could be true in exactly two interpretations. I will leave figuring them out to the exercises. The following list exhausts the possible truth conditions of any compound sentence ⟨A⊗B⟩:

- 1 way in which the compound sentence is true in *no* interpretation.
- 4 ways in which the compound sentence is true in exactly one interpretation.
- 6 ways in which the compound sentence is true in exactly two interpretations.
- 4 ways in which the compound sentence is true in exactly three interpretations.
- 1 way in which a compound sentence is true in all four interpretations.

Altogether, we have 16 ways the truth condition of a compound sentence ⟨A⊗B⟩ can be.

So, given just A and B as atomic sentences, there are only sixteen different things you could say by compound sentences.

Notice that the number of compound sentences must be much larger than 16 as can be seen from the following simple sequence of compound sentences:

1. ⟨A and B⟩.
2. ⟨B and ⟨A and B⟩⟩.
3. ⟨A and ⟨B and ⟨A and B⟩⟩⟩.
4. ⟨B and ⟨A and ⟨B and ⟨A and B⟩⟩⟩⟩.
5. ⟨A and ⟨B and ⟨A and ⟨B and ⟨A and B⟩⟩⟩⟩⟩.

⋮

You can continue the sequence ad infinitum which means that even with just two atomic sentences, there are infinitely many compound sentences. But

there are only 16 different things you can say using those infinitely many compound sentences—all we've got is infinitely many ways of saying 16 things.

Of course, there is nothing special about the particular atomic sentences A and B: given two atomic sentences, whatever they are, there are only 16 different things that can be said by compound sentences constructed out of them.

Chapter 2

Formal Languages

2.1 Introducing Formal Languages

So far, we have been discussing English sentences using English. One of the great strengths of natural languages is their expressive power and versatility that enable us to talk about a language using that very language. However, the long history of the study of logic, languages, and our reasoning abilities shows that this versatility is also a source of confusion and avoidable errors. Systematic treatment is easier if we introduce an artificial language and talk about the features of that language.

The language that we talk about is the *object language*. The language we use to talk about the object language is the *meta-language*. So far, our object language has been the same language as the meta-language: we have been using English to talk about English sentences. We will change the object language to a new artificial language. Our meta-language remains English—no need to learn a new language to continue reading—but the object language will be an artificial language whose features we stipulate to suit our needs and can control with great precision. The aim is to construct an artificial language that can do everything, or nearly everything, by way of expressing logical relations that we can in English. But, unlike English, our artificial language is incapable of doing much else.

Let us start. Let us call our artificial language $\mathcal{L}_S$ (that's a cursive 'L' as in 'language' with a subscript 'S' as in 'simple'). $\mathcal{L}_S$ has two expressions, A and B, that both express sentences. The following tables tell us what they mean:

snow is white	A
T	T
F	F

the moon is made of cheese	B
T	T
F	F

These tables state the truth conditions of A and B. As you can see, when a speaker of $\mathcal{L}_S$ uses A to say something, what they say is true if snow is white, and what they say is false otherwise. If the speaker uses B to say something, what they say is true if the moon is made of cheese, and what they say is false otherwise. So you could translate A into the English sentence 'snow is white' and translate B into 'the moon is made of cheese'.

Here is something *very* important to keep in mind. If what I have told you about $\mathcal{L}_S$ exhausts the features of $\mathcal{L}_S$, a monolingual speaker of $\mathcal{L}_S$ could not say what I just said in the previous paragraph. In order to say what I just said I needed many more sentences than just A and B or their English translations. $\mathcal{L}_S$ does not have any of those sentences available.

For the same reason, a monolingual speaker of $\mathcal{L}_S$ could not give us the table describing when A and B are true. We are using English as our meta-language and I am relying on your knowledge of English even if it is not your first language. Because English is so much more powerful than $\mathcal{L}_S$, we can describe features of $\mathcal{L}_S$ in ways that native monolingual speakers of $\mathcal{L}_S$ could not.

So far, $\mathcal{L}_S$ is not a very useful language. Let us expand it a bit. We want it to be able to form compound sentences. To that end, we will introduce the following three symbols: $\neg$, $\wedge$, $\vee$. Additionally, we also have brackets: (and). These symbols can be used to form the following compound sentences:

1. $\neg A$
2. $(A \wedge B)$
3. $(A \vee B)$

These are the only compound sentences you can form in $\mathcal{L}_S$ so far. What do these sentences express? We can specify that by stipulating the truth conditions of these compound sentences. Let's do this one by one. Here is the first.

A	$\neg A$
T	F
F	T

This tells us that $\neg A$ is true iff. A is false: it does not matter whether or not B is true. Since we know from above that A is false iff. it is not the case that snow is white, $\neg A$ is true iff. it is not the case that snow is white. So, $\neg A$ is the denial of A. You could translate $\neg A$ as 'it is not the case that snow is white'. Let us call the $\neg$ symbol the *negation* sign.

Let us move to the next compound sentence, $(A \wedge B)$. Its truth condition is given by:

A	B	$(A \wedge B)$
T	T	T
F	T	F
T	F	F
F	F	F

As you can see, the compound sentence $(A \wedge B)$ is true if and only if A and B are both true. So one way of translating $(A \wedge B)$ into English would be 'A and B' (i.e., 'snow is white and the moon is made of cheese'). Let us call the $\wedge$ sign the *conjunction* sign. The sentences flanking the sign are called *conjuncts*.

Here is the last compound sentence: $(A \vee B)$:

A	B	$(A \vee B)$
T	T	T
F	T	T
T	F	T
F	F	F

So $(A \vee B)$ is true if and only if at least one of A or B is true. Let us call the $\vee$ sign the *disjunction* sign. The sentences flanking the sign are called *disjuncts*.

We now have expanded $\mathcal{L}_S$ to be able to form a few compound sentences. You might have noticed that they work an awful lot like the logical connectives 'it is not the case that', 'and', and 'or' of the English language that we discussed in the previous chapter. But they are not quite the same as those logical connectives of English yet.

Notice, for instance, that given the specifications so far, a monolingual speaker can deny A in their language but cannot deny B—the list of compound sentences of $\mathcal{L}_S$ above does not contain one for denying B. Nor is it yet possible to deny the conjunction $(A \wedge B)$. So we need a bit more work to have a useful language.

2.2 Truth Tables

We noted above that $\mathcal{L}_S$ does not have the ability to express the denial of B even though it can deny A by the compound sentence $\neg A$. It is natural to introduce a way of denying B by allowing another compound sentence: $\neg B$. But notice that merely allowing that $\neg B$ is a compound sentence of $\mathcal{L}_S$ does not ensure that it is the denial of B. We could have a language in which a prefix like $\neg$ functions differently depending on what it is attached to—think of the prefix 'ex' used in English; sometimes it means something like 'outside', other times it means something like 'previous'. So just as we stipulated the truth conditions of $\neg A$, we need to stipulate the truth conditions for $\neg B$. That's not too difficult: take the above table for $\neg A$ and replace each A with B. But there is a problem. We want to be able to deny any arbitrary sentence. If we had to state the truth conditions for each and every sentence, that will get old very quickly, and we could not finish the task since there are infinitely many possible compound sentences: e.g., we want to be able to deny a denial, deny the denial of a denial, etc. We need a concise way of stating the truth conditions for the infinitely many possible sentences we could form by prefixing them with the negation sign.

Let us not forget that $\mathcal{L}_S$ is our own creation and we get to control how it works. So let us just say that prefixing any sentence of $\mathcal{L}_S$ with the $\neg$ sign results in a compound sentence which is the negation of the first. We can represent this using the following table:

s	$\neg s$
T	F
F	T

This is called the *truth table* for negation. Notice the lower case s. It does not stand for any particular sentence. Rather, it is a sentence *variable* into which you can substitute any sentence of $\mathcal{L}_S$. So the table is like a template: you can see how the truth value of the sentence $\neg A$ depends on the truth value of A by substituting A for s everywhere in the above table. And you can see how the truth value of the sentence $\neg B$ depends on the truth value of B by substituting B into s everywhere in the table. That is, the truth table for negation can help you figure out the truth conditions for sentences like $\neg A$ and $\neg B$. Similarly, if you want to see how the truth value of $\neg(A \wedge B)$ depends on the truth value of $(A \wedge B)$, substitute $(A \wedge B)$ for s everywhere, like this:

$(A \wedge B)$	$\neg(A \wedge B)$
T	F
F	T

Notice that this table does not give us the full list of interpretations: given two atomic sentence A and B, there must be four interpretations. We will look at expanding this to a full specification of the truth condition of $\neg(A \wedge B)$ that covers all interpretations in the exercises.

A *truth table* tells us how a particular connective like $\neg$ works by giving us a template for generating tables that tell us how the truth values of a complex sentence using that connective depend on the truth values of its component sentences. I have just used the word 'connective' as in 'logical connective'. After all, given the truth table, the $\neg$ sign in $\mathcal{L}_S$ works just like the English 'it is not the case that': it can be prefixed to any sentence to form a denial of that sentence, and the truth value of the resulting sentence is a function of the truth value of the original sentence.

Let us now move to the other two connectives. Here is the truth table for conjunction:

s_1	s_2	$(s_1 \wedge s_2)$
T	T	T
F	T	F
T	F	F
F	F	F

Because a conjunction takes two sentences to form a new one, we indicate that by the subscripts 1 and 2. The subscripts help you keep track of what you substitute into what: all occurrences of s_1 must be substituted by the same sentence, and ditto for s_2. But this does *not* mean that you are not allowed to substitute the same sentence into both s_1 and s_2.

Substituting A into s_1 and B into s_2 tells us how the value of $A \wedge B$ depends on the truth values of A and B, and the result of the substitution is identical to the truth conditions of 'A and B' that we stated previously. But the truth table also tells us a lot more. Given any two sentences, the table tells us how the truth value of the compound sentence generated by connecting the two with the $\wedge$ sign depends on the truth values of the two component sentences.

Finally, here is the truth table for disjunction:

s_1	s_2	$(s_1 \vee s_2)$
T	T	T
F	T	T
T	F	T
F	F	F

We now have everything we need to figure out the truth-conditions for any sentence of the forms: $\neg s$, $(s_1 \wedge s_2)$, $(s_1 \vee s_2)$. The sentences s, s_1, s_2 can all be compound sentences themselves. What about something you get by prefixing the $\wedge$ sign to a sentence, like $(\wedge A)$? Well, $\mathcal{L}_S$ is our creation so let us declare that there are no grammatically correct sentences in $\mathcal{L}_S$ apart from the atomic sentences A and B and the three types of compound sentences dealt with by the three truth tables. Instead of calling these sentences grammatically correct, logicians call them *well-formed formulas (wff)*.

Let me remind you that the characterizations of $\mathcal{L}_S$ that we are giving are all given in the meta-language (English, in the current case). A monolingual speaker of $\mathcal{L}_S$ (even the expanded version) could not give the characterizations we have just been giving since $\mathcal{L}_S$ itself does not have the tools that we are using in giving the characterizations. Native speakers of a language often have trouble spelling out the workings of their own language—mono-lingual speakers of $\mathcal{L}_S$ are an extreme version of that.

Going forward, in reading sentences of $\mathcal{L}_S$, you can pronounce the symbols $\neg$, $\vee$, and $\wedge$ as 'not', 'or', and 'and' just because it makes life easier. But keep in mind that those symbols are expressions of $\mathcal{L}_S$ and not words of English. We assign sounds to them for convenience. If your native tongue is not English, you might very well prefer to assign different sounds. It does not matter so long as we are clear how $\mathcal{L}_S$ works and that is given by their truth tables just discussed.

> Those of you familiar with the mathematical concept of function will notice that a truth table represents a function mapping ordered n-tuples of truth values on the left side of the vertical line onto truth values on the right side. That is why the particular kind of connectives we have been interested in are called 'truth-*functional*.'

2.3 The Language of Sentential Logic

So far, our language $\mathcal{L}_S$ has had only two atomic sentences. That made reasoning about them easy but it is also extremely limiting: there are at most 16 different things we can say about the world using truth-functional compound sentences in a language with only two atomic sentences. Let us therefore allow many more atomic sentences. Let us define:

Definition. A formal language $\mathcal{L}$ of sentential logic has the following features:

1. There are countably many *atomic* sentences.
2. If s is a sentence of $\mathcal{L}$, so is $\neg s$.
3. If s_1 and s_2 are sentences of $\mathcal{L}$, so is $(s_1 \wedge s_2)$.
4. If s_1 and s_2 are sentences of $\mathcal{L}$, so is $(s_1 \vee s_2)$.
5. There are no other sentences of $\mathcal{L}$.
6. Each atomic sentence has a truth value (either true or false).
7. Given any sentence s of $\mathcal{L}$, $\neg s$ is true iff. s is false.
8. Given any sentences s_1 and s_2 of $\mathcal{L}$, $(s_1 \wedge s_2)$ is true iff. s_1 and s_2 are both true.
9. Given any sentences s_1 and s_2 of $\mathcal{L}$, $(s_1 \vee s_2)$ is true iff. at least one of s_1 and s_2 is true.

That's it. A few remarks:

To say that $\mathcal{L}$ has countably many atomic sentences is to say that you can count the atomic sentences in the usual ways (the first, the second, etc.) all the way up to infinity. That is, there could be as many atomic sentences as there are natural numbers (or positive integers). But no more.

You may have noticed the use of the indefinite article in the definition: '*a* formal language $\mathcal{L}$ ...'. The above definition does not define any particular language. Rather, it states the conditions something must satisfy to count as a formal language of sentential logic. There can be many different formal languages that fit the bill. They can differ in the number of atomic sentences and the meanings of them. For example, $\mathcal{L}_S$ is a formal language as defined here. It just has two atomic sentences.

The first five conditions specify the *syntax* of $\mathcal{L}$, the others the *semantics*. The syntax tells us how the sentences of $\mathcal{L}$ look, the semantics tells us about the meanings of the sentences.

A sentence as defined above is often called a *well-formed formula (wff)*.

Notice how compound sentences are built. There are three ways a compound sentence can be formed:

i. prefix a sentence with the negation sign;

ii. connect two sentences with the conjunction sign;

iii. connect two sentences with the disjunction sign.

Since these are the only ways, any compound sentence must either be a negation of another, a conjunction of two sentences, or a disjunction of two sentences. We will say that in the first case the *main connective* is negation, in the second case the main connective is conjunction, and in the last case the main connective is disjunction. The main connective tells you what the last step of construction was to get to the compound sentence starting from atomic sentences.

The use of brackets is mandatory as given by the definitions. However, to improve readability, you are allowed to do the following:

- If the very first symbol of a sentence is the opening bracket and its matching closing bracket is the very last symbol of the sentence, you may omit that particular pair of brackets.
- you may vary them in size as well as shape (e.g., you might use square brackets, curly brackets, etc.) so long as you match both the size and shape of opening and closing brackets.

Thus, for example, you may write

$$\big[P \vee \big(Q \wedge R\big)\big] \wedge W$$

instead of

$$\big(\big(P \vee \big(Q \wedge R\big)\big) \wedge W\big)$$

But you may *not* write $\neg A \wedge B$ instead of $\neg(A \wedge B)$ (notice that if the main connective is the negation, the very first symbol of the sentence is $\neg$ rather than an opening bracket).

Sometimes, it helps readability to add brackets so we will allow adding extra brackets: if s is a sentence, so is (s). E.g., you may write $\neg(\neg P)$ instead of $\neg\neg P$—bracketing off negations probably is the main useful case of adding extra brackets.

In this class, we will use upper case Roman Alphabet letters like A, B, C to stand for sentences. We will use lower case letters like p, q, r, possibly with subscripts as *sentential variables*. Sentential variables can be used to formulate *sentence templates* like $(s_1 \vee s_2)$ but also more complex ones like:

$$\neg\big[\neg s_1 \wedge \big(s_1 \vee s_2\big)\big] \vee s_2$$

You can get actual sentences by plugging sentences into the sentence variables with the rule that different occurrences of the same sentence variable must be replaced by the same sentence. We often will be discussing sentence templates because that allows us to discuss all sentences that fit the form of the template at once.

There are various alternative symbols you will find in logic text books. For negation, instead of $\neg$ you might see $-$ or $\sim$, or sometimes a bar above the sentence letter. Instead of $\wedge$ you might see $\&$ or a dot $(\cdot)$ like a multiplication sign. And instead of $\vee$, you might see $+$. Some have more elaborate conventions to reduce the need for brackets. Etc. These differences are largely a matter of taste and inertia and do not indicate any difference in content. I am using the symbols that I happen to have learned first.

2.4 Figuring Out Truth Tables: Two Atomic Sentences

Let's see how we can figure out the truth table of an arbitrary compound sentence of $\mathcal{L}$.

Any compound sentence of $\mathcal{L}$ is constructed out of atomic sentences through procedures captured by the definition of $\mathcal{L}$. Figuring out the truth table of any given sentence starts with figuring out how they are constructed. They are very much like Lego creations. You have certain basic pieces which you can connect to form ever more complex compound sentences. Just as with Lego creations, you can figure out how a compound sentence was constructed by taking it apart systematically.

Consider the following sentence:

$$(P \wedge Q) \wedge (\neg P \vee Q)$$

P and Q are atomic sentences. How is this compound sentence constructed in accordance with the rule governing the compound sentences of $\mathcal{L}$? We can reverse engineer.

The main connective of the sentence is $\wedge$. That means the final step in the construction was connecting the following two pieces with $\wedge$:

- $P \wedge Q$
- $\neg P \vee Q$

These are themselves compound sentences. How are they constructed? The former is a conjunction of two atomic sentences:

- P
- Q

The latter is a disjunction of:

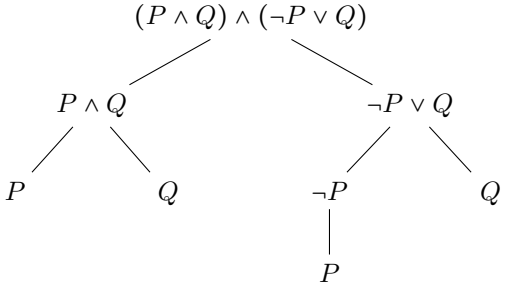
- $\neg P$

- Q

The last is an atomic sentence, but the former here is a compound sentence whose main connective is negation (¬) and it negates:

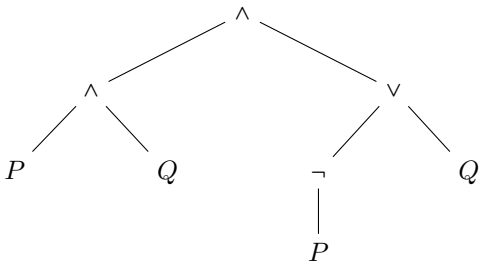
- P

We can represent this structure in tree form:



At the top you have the whole sentence, and as you go down you see the component sentences as branches. As you can see, each *node* has a maximum of two branches and eventually everything terminates in an atomic sentence. Since atomic sentences have no further branches, they are called *leaf nodes*.

In linguistics and computer science you often see what we just outlined in a more simplified form that only shows the connectives and atomic sentences. Such a tree is known as a *syntax tree* (or parse tree):



We can use this to figure out the truth conditions of the whole sentence. The point to notice is that the truth value of a compound sentence depends on the truth value of its component sentences. And we can figure out the truth values of the component sentences step-by-step starting with the truth values of the atomic sentences. We can represent that in table form:

P	Q	$P \wedge Q$	$\neg P$	$\neg P \vee Q$	$(P \wedge Q) \wedge (\neg P \vee Q)$
T	*T*	*T*	*F*	*T*	*T*
F	*T*	*F*	*T*	*T*	*F*
T	*F*	*F*	*F*	*F*	*F*
F	*F*	*F*	*T*	*T*	*F*

On the top row, from left to right, we have the atomic sentences and a sequence of compound sentences corresponding to the way you can build up to the target compound sentence. Under the horizontal line, again from left to right, you have first the four possible interpretations, and then the truth values for the sequence of compound sentences up to the target sentence. We figure out the truth values for each compound sentence by consulting the truth tables for the main connective of the sentence (in this case, first negation, then conjunction, then disjunction, and finally conjunction again).

E.g., take the first row: P and Q are both true on the first row. So the truth table of conjunction tells us that $P \wedge Q$ is true. The truth table for negation tells us that $\neg P$ is false, and the truth table for disjunction tells us that $\neg P \vee Q$ is true. And we get that $(P \wedge Q) \wedge (\neg P \vee Q)$ is true because its conjuncts are both true.

Take the second row: P is false and Q is true on the second row. That means $P \wedge Q$ is false because the first conjunct is false. $\neg P \vee Q$, however, is true since both disjuncts are true. The whole sentence $(P \wedge Q) \wedge (\neg P \vee Q)$ is false because only one of its conjuncts is true.

As you can see, the truth condition for the whole sentence is identical to the truth condition for $P \wedge Q$. When two sentences have the same truth condition they are said to be *logically equivalent*. So $P \wedge Q$ and $(P \wedge Q) \wedge (\neg P \vee Q)$ are logically equivalent.

More generally, in order to figure out the truth table of a compound sentence, start with the interpretations of the atomic sentences, and then work your way up step-by-step through increasingly complex compound sentences until you reach the target sentence. You often will have some options concerning the

order of columns. But always make sure that you can tell what goes into a given column by consulting only columns to the left of it.

2.5 Figuring Out Truth Tables: More Than Two Atomic Sentences

Let's take a look at a sentence involving more than two atomic sentences. The first thing we need to make sure is that we list all the interpretations. How can we do that?

Suppose you have figured out the table of all interpretations for *n* atomic sentences. The following is a procedure for generating the table of all interpretations for *n+1* sentences:

1. Generate the table of interpretations for *n* atomic sentences. Call the rows of this table the *original rows*.
2. Extend the table downwards by duplicating the table for *n* atomic sentences under it. Call the added rows the *new rows*.
3. Add a column to the right of table as follows: in the original rows add *T*s; in the new rows add *F*s.

Here is an example. If there is only one atomic sentence s_1, the following is the table of all interpretations.

s_1
T
F

We now extend the table downward by duplicating the original table. The original is shaded so you can see better what is going on:

s_1
T
F
T
F

We add a column to the right writing *T*s in the rows belonging to the original table, and *F*s in the other rows which results in the table of interpretations for two atomic sentences s_1 and s_2:

s_1	s_2
T	*T*
F	*T*
T	*F*
F	*F*

We can now generate the table for three atomic sentences by reapplying the procedure. Here is how it looks. The original table is shaded:

step 1.

s_1	s_2
T	*T*
F	*T*
T	*F*
F	*F*

step 2.

s_1	s_2
T	*T*
F	*T*
T	*F*
F	*F*
T	*T*
F	*T*
T	*F*
F	*F*

step 3.

s_1	s_2	s_3
T	*T*	*T*
F	*T*	*T*
T	*F*	*T*
F	*F*	*T*
T	*T*	*F*
F	*T*	*F*
T	*F*	*F*
F	*F*	*F*

And here is how we get the table of interpretations for four atomic sentences:

step 1:

s_1	s_2	s_3
T	*T*	*T*
F	*T*	*T*
T	*F*	*T*
F	*F*	*T*
T	*T*	*F*
F	*T*	*F*
T	*F*	*F*
F	*F*	*F*

step 2:

s_1	s_2	s_3
T	*T*	*T*
F	*T*	*T*
T	*F*	*T*
F	*F*	*T*
T	*T*	*F*
F	*T*	*F*
T	*F*	*F*
F	*F*	*F*
T	*T*	*T*
F	*T*	*T*
T	*F*	*T*
F	*F*	*T*
T	*T*	*F*
F	*T*	*F*
T	*F*	*F*
F	*F*	*F*

step 3:

s_1	s_2	s_3	s_4
T	*T*	*T*	*T*
F	*T*	*T*	*T*
T	*F*	*T*	*T*
F	*F*	*T*	*T*
T	*T*	*F*	*T*
F	*T*	*F*	*T*
T	*F*	*F*	*T*
F	*F*	*F*	*T*
T	*T*	*T*	*F*
F	*T*	*T*	*F*
T	*F*	*T*	*F*
F	*F*	*T*	*F*
T	*T*	*F*	*F*
F	*T*	*F*	*F*
T	*F*	*F*	*F*
F	*F*	*F*	*F*

We can continue this procedure to get the table of interpretations for any number of atomic sentences. I will leave explaining why this procedure will generate the complete table of interpretations for any finite number of atomic sentences as an exercise.

Once we have the table of interpretations, the rest of the truth table for n atomic sentences gets constructed in the same way as the one for two atomic sentences. You just have to work through more rows. Here is an example which is the truth table for $(P \wedge Q) \wedge R$:

P	Q	R	$P \wedge Q$	$\big(P \wedge Q\big) \wedge R$
T	T	T	T	T
F	T	T	F	F
T	F	T	F	F
F	F	T	F	F
T	T	F	T	F
F	T	F	F	F
T	F	F	F	F
F	F	F	F	F

Here is another example for $\neg\big(P \vee R\big) \vee \big(Q \wedge S\big)$:

P	Q	R	S	$P \vee R$	$\neg\big(P \vee R\big)$	$Q \wedge S$	$\neg\big(P \vee R\big) \vee \big(Q \wedge S\big)$
T	T	T	T	T	F	T	T
F	T	T	T	T	F	T	T
T	F	T	T	T	F	F	F
F	F	T	T	T	F	F	F
T	T	F	T	T	F	T	T
F	T	F	T	F	T	T	T
T	F	F	T	T	F	F	F
F	F	F	T	F	T	F	T
T	T	T	F	T	F	F	F
F	T	T	F	T	F	F	F
T	F	T	F	T	F	F	F
F	F	T	F	T	F	F	F
T	T	F	F	T	F	F	F
F	T	F	F	F	T	F	T
T	F	F	F	T	F	F	F
F	F	F	F	F	T	F	T

You will find that many textbooks have truth tables with the columns under the atomic sentences in reverse order from ours: rather than the first atomic sentence having a column of alternating T's and F's, it's the last atomic sentence that has such a column. I don't use that format—I realize that makes it a bit inconvenient to consult other textbooks (for me too!)—because our way of ordering interpretations will make things easier to see in Section 2.8 as well as Section 4.8. I should note that there is at least one influential book on logic that uses our ordering: Hunter (1996).

You might also notice that other logic textbooks often use ways of writing down truth tables that are more compact. Their main virtue is compactness, but they come with a steeper learning curve because 'reading' them is a bit trickier and you need to keep more in your head during construction. Apart from these matters, those other ways of doing truth tables are identical to what you see here.

2.6 Tautologies and Contradiction

As we have seen earlier, there are sentences that are true in all interpretations (recall section 1.5). A sentence that is true in all interpretations is called a *tautology*. Since the world must be such that one of the possible interpretations correctly describes the world, a tautology is a sentence that is true no matter what the world is like. We do not need to do any research into what the world is like to see that a tautology is true.

Some other sentences are false no matter what. Such sentences are known as *contradictions*. We do not need to do any empirical research to see that they are false.

Sentences that are true in some, but not all, interpretations are *contingent* sentences. Figuring out whether or not a contingent sentence is true requires investigating what the world is like.

A number of sentences are *consistent* with each other iff. there is an interpretation in which all the sentences in the set are true. Sentences are *inconsistent* with each other iff. there is no interpretation in which all the sentences are true. Derivatively, we will also speak of an individual sentence's being consistent: a sentence is consistent iff. there is an interpretation in which it is true—so both tautologies and contingent sentences are consistent. A sentence is inconsistent iff. it is not consistent. I.e., an inconsistent sentence is a contradiction.

To make life easier, we will also say that sentence templates are tautologies. A sentence template is a tautology iff. sentences that fit the template are tautologies. Mutatis mutandis for saying that a sentence template is a contradiction, contingent, and consistent.

Some tautologies and contradictions are obviously so, like $P \vee \neg P$ and $P \wedge \neg P$. But there are other tautologies that are much harder to show that they are in fact tautologies. If you recall our earlier discussion in chapter 1, given two atomic sentences, compound sentences must have one of 16 possible truth conditions. One of the possible truth conditions is that a sentence is true no matter what, another is that a sentence is false no matter what. This means that we should expect there to be infinitely many tautologies, as well as infinitely many contradictions. Most of them will be too complex for us to see whether they are tautologies/contradictions without serious work. For instance, the following are all tautologies:

- $\neg\big[(Y \wedge Z) \wedge Y\big] \vee \Big((P \wedge L) \vee \big\{\big[Z \vee (Z \wedge W)\big] \vee \big[(Y \wedge P) \vee (W \wedge W)\big]\big\}\Big)$
- $\big\{\big[\neg Z \vee (Z \wedge P)\big] \vee (R \vee Z)\big\} \vee \big\{\big[T \vee (Q \vee Q)\big] \vee \big[(P \vee S) \vee (M \wedge Y)\big]\big\}$
- $\big\{\big[(W \vee P) \wedge R\big] \vee (Z \vee W)\big\} \vee \big[\neg(W \wedge M) \vee \neg(S \vee L)\big]$
- $\Big(M \vee \big\{(Q \wedge Q) \wedge \big[(Y \vee Q) \wedge (W \vee L)\big]\big\}\Big) \vee \big\{T \vee \neg\big[T \wedge (T \vee S)\big]\big\}$
- $\neg\big\{\big[(P \wedge T) \vee (P \vee S)\big] \wedge \big[R \wedge (Q \wedge Y)\big]\big\} \vee \big\{\big[(R \vee P) \wedge R\big] \vee R\big\}$
- $\neg\Big(\neg\big[(L \vee Z) \vee S\big] \wedge \big\{(T \vee P) \wedge \big[(L \wedge T) \wedge (Z \vee W)\big]\big\}\Big)$
- $\Big(\neg R \vee \big\{\big[(Q \wedge T) \vee Q\big] \vee W\big\}\Big) \vee \big\{\big[(Z \vee Z) \wedge (P \vee P)\big] \vee R\big\}$
- $Y \vee \Big(\big\{\big[M \vee (R \wedge S)\big] \wedge S\big\} \vee \big\{\big[(W \wedge W) \wedge (P \wedge L)\big] \vee \neg Y\big\}\Big)$
- $\Big(\big\{\big[(Y \vee Z) \vee R\big] \vee \neg(Y \wedge W)\big\} \vee (Z \wedge M)\Big) \vee \big[Q \wedge (P \wedge R)\big]$
- $\neg\big[(T \wedge Z) \wedge (S \wedge T)\big] \vee \big\{\big[(Z \wedge T) \wedge (P \vee S)\big] \vee (M \wedge P)\big\}$
- $\Big[\big[(S \vee P) \wedge (L \vee P)\big] \vee \Big((P \vee R) \vee \big\{W \vee \big[\neg Z \vee (S \vee R)\big]\big\}\Big)\Big] \vee (Z \vee T)$
- $\big[(Q \vee Z) \vee \neg(T \wedge R)\big] \vee \big\{\big[(Q \wedge R) \vee (P \wedge Y)\big] \vee \big[R \vee (L \vee W)\big]\big\}$
- $\big\{Z \vee \big[(R \wedge W) \vee (M \vee Z)\big]\big\} \vee \big\{\neg\big[Y \wedge (Z \vee R)\big] \vee (Z \vee R)\big\}$
- $P \vee \Big(\big\{\big[(S \vee W) \vee \neg W\big] \vee (L \vee Y)\big\} \vee \big[(L \wedge T) \wedge (W \vee Q)\big]\Big)$

You could try working through truth tables for these but there are limits to that. With just a small increase in complexity, the inspection of truth tables becomes humanly impossible. But the way truth tables are constructed makes them ideally suited for checking by machines like the ubiquitous digital computers—that's how I know that the above are all tautologies.

2.7 Conditionals

So far we have had only three connectives: $\wedge$, $\vee$, and $\neg$. However, restricting ourselves to these three can be very cumbersome. One particular connective that we use in ordinary English but has not found its way into our $\mathcal{L}$ yet is the connective that appears in construction of the form 'if ..., then ...'. Our own discussion in the meta-language (English) so far would have been extremely difficult to conduct without this type of formulation. The *conditional*, as it is known, is a very basic part of our vocabulary and it would be good to have its analogue in $\mathcal{L}$. So without further ado let's introduce such a connective.

We introduce the logical connective $\supset$ (the horseshoe): if s_1 and s_2 are sentences of $\mathcal{L}$, so is $(s_1 \supset s_2)$. You can read this variously as 'if s_1, then s_2', 's_1 implies s_2', 's_1 only if s_2', or 's_1 materially implies s_2'. We call what's left of the $\supset$ symbol the *antecedent*, what's right of the symbol the *consequent*. A sentence of the form $s_1 \supset s_2$ is often known as the 'conditional' or as the 'material implication.'

If $\supset$ is a logical connective, the truth value of a sentence $(s_1 \supset s_2)$ must depend on the truth values of s_1 and s_2 only. What is the truth table of $\supset$? We want $\supset$ to reflect how the 'if, then' construction works in English. Here are some features of this construction:

- Given the truth of ⟨if A, then B⟩ and the truth of A, it follows that B. So we want the truth table of the conditional to have the following row:

s_1	s_2	$s_1 \supset s_2$
T	T	T

- Given that A is true and B is false, it follows that it is not true that ⟨if A, then B⟩. So we want the following row:

s_1	s_2	$s_1 \supset s_2$
T	F	F

- Suppose ⟨if A, then B⟩ is true but that A is false. What follows about B? Nothing, since the conditional only tells us what is the case if A is true, but that is silent about what is the case if A is false. So we want the following rows:

s_1	s_2	$s_1 \supset s_2$
F	T	T
F	F	T

So we already know all we need to know. The truth table for the conditional $\supset$ is:

s_1	s_2	$(s_1 \supset s_2)$
T	T	T
F	T	T
T	F	F
F	F	T

This may be surprising, but it does a good job capturing how we reason using 'if ..., then...' constructions. After all, we extracted the truth table from some fairly trivial observations about such constructions.

The connective $\supset$ is introduced for convenience only. As we will see in the exercises, we can say what $s_1 \supset s_2$ says using only the previously available three connectives. But it is nevertheless extremely convenient to have, especially for investigating arguments which is the topic of our next chapter.

Here is the updated definition of a formal language of sentential logic that includes $\supset$:

Definition. A formal language $\mathcal{L}$ of sentential logic has the following features:

1. There are countably many *atomic* sentences.
2. If s is a sentence of $\mathcal{L}$, so is $\neg s$.
3. If s_1 and s_2 are sentences of $\mathcal{L}$, so is $(s_1 \wedge s_2)$.
4. If s_1 and s_2 are sentences of $\mathcal{L}$, so is $(s_1 \vee s_2)$.
5. If s_1 and s_2 are sentences of $\mathcal{L}$, so is $(s_1 \supset s_2)$.
6. There are no other sentences of $\mathcal{L}$.
7. Each atomic sentence has a truth value (either true or false).
8. Given any sentence s of $\mathcal{L}$, $\neg s$ is true iff. s is false.

9. Given any sentences s_1 and s_2 of $\mathcal{L}$, $(s_1 \wedge s_2)$ is true iff. s_1 and s_2 are both true.
10. Given any sentences s_1 and s_2 of $\mathcal{L}$, $(s_1 \vee s_2)$ is true iff. at least one of s_1 and s_2 is true.
11. Given any sentences s_1 and s_2 of $\mathcal{L}$, $(s_1 \supset s_2)$ is true iff. s_1 is false or s_2 is true.

2.8 Enumerating Sentences of $\mathcal{L}$

Suppose you are asked to come up with a procedure for writing down a list of sentences of $\mathcal{L}$ one by one such that: given any particular sentence S of finite length of $\mathcal{L}$, S will eventually appear on the list. Is there such a procedure? (If you are familiar with a bit of computer programming, the task is to write a program that outputs *all* finite-length sentences of a given $\mathcal{L}$ one by one.) The task is not trivial. You might, for instance, try first listing all the sentences that only use $\wedge$, then listing all the sentences that use only $\vee$, etc. But on such a procedure you will never see $A \vee B$ appear on the list because you will never finish listing all the sentences that use only $\wedge$. So can we construct a procedure which ensures that any arbitrary sentence of $\mathcal{L}$ will appear on the list eventually?

The short answer is Yes. Even though any formal language $\mathcal{L}$ has infinitely many sentences, it is possible to specify a procedure for generating a list of the sentences of $\mathcal{L}$ such that any arbitrary sentence of $\mathcal{L}$ will eventually appear on the list. Since we could number the items on the list in the usual way (the first one is numbered 1, the second one is numbered 2, etc.), this means we can *enumerate* all the sentences of $\mathcal{L}$. When items can be enumerated in this way, the items are *countable*—because we can count them: 'here is the first', 'here is the second', etc. So the sentences of any $\mathcal{L}$ are countable. This is a very important point. Let me sketch a proof that the sentences are indeed countable.

2.8.1 Countability of fractions

Let's start with a somewhat simpler question. A positive fraction is a number of the form m/n where m and n are natural numbers (whole numbers from 1 up). Is there a procedure for listing all fractions such that any fraction will eventually appear on the list?

You might try this: First list all the fractions of the form 1/n for all natural numbers n. Then list all the fractions of the form 2/n. Then list all the fractions of the form 3/n. Etc. But this will not work. You will never finish listing all the fractions of the form 1/n, so the fraction 2/3 will never appear on the list as it is produced. Can there be a procedure for enumerating all positive fractions?

The very idea of enumerating all the fractions might seem puzzling. After all, between any two natural numbers, there are infinitely many fractions. Not

only that, between *any* two fractions, there are infinitely many more. But there is a surprisingly simple way shown by Georg Cantor (1895). Consider Figure 2.1 below. It lays out all the fractions on an array. The first row lists all the fractions with enumerator 1 with the denominator increasing by one from one column to the next. The second row lists all the fractions with enumerator 2. Etc. You can put all the fractions on a single string, as it were, by walking along the path indicated by the arrows starting from 1/1 and continuing in accordance with that pattern. When you straighten the string, you get the sequence of fraction that looks like this: 1/1, 2/1, 1/2, 1/3, 2/2, 3/1, 4/1, 3/2, ... Any possible fraction will appear on the list because any fraction is somewhere on the array and the zigzagging path will hit any fraction eventually. (What we are doing is list for each natural number larger than 1 those fractions whose denominator and enumerator add up to that number. We keep as separate items fractions that are mathematically equal—like 2/3 and 4/6—because it will make our lives easier later.)

1/1		1/2	→	1/3		1/4	→	1/5		1/6		...
↓	↗		↙		↗		↙					
2/1		2/2		2/3		2/4		2/5		2/6		...
	↙		↗		↙							
3/1		3/2		3/3		3/4		3/5		3/6		...
↓	↗		↙									
4/1		4/2		4/3		4/4		4/5		4/6		...
	↙											
5/1		5/2		5/3		5/4		5/5		5/6		...
↓	↗											
6/1		6/2		6/3		6/4		6/5		6/6		...
7/1		7/2		7/3		7/4		7/5		7/6		...
⋮		⋮		⋮		⋮		⋮		⋮		

Figure 2.1: Procedure for enumerating all positive fractions.

So fractions are countable: the first is 1/1, the second is 2/1, the third is 1/2, etc. One way of thinking about what we are doing is that we are assigning ID numbers to each fraction where each ID number is a natural number. Even though one might feel that there are many more fractions than there are natural numbers, we can in fact give each and every fraction its own unique ID

number without exhausting the pool of ID numbers.

2.8.2 Countability of sentences

Suppose $\mathcal{L}$ has infinitely many atomic sentences A_1, A_2, A_3, ... We want to enumerate all sentences of $\mathcal{L}$. How could we do that? We will proceed in stages.

Let us first classify the sentences of $\mathcal{L}$. We will call our atomic sentences class-0 sentences. Class-1 sentences are all class-0 sentences and all complex sentences that can be formed out of class-0 sentences by applying a single connective once. Class-3 sentences are all class-1 sentences and all complex sentences that can be formed out of class-1 sentences by applying a single connective once. More generally:

Class-0 sentences are atomic sentences.

Class-(n+1) sentences are all class-n sentences plus all sentences that can be formed out of at most two class-n sentences by applying a single connective once.

Notice that any sentence of $\mathcal{L}$ must belong to some class-n (more precisely, there is an n such that the sentence belongs to any class-m such that m≥n). We will now show that for each n, class-n sentences are countable.

Let's start with class-0 sentences. They are the atomic sentences of $\mathcal{L}$ and by the definition of formal languages, class-0 sentences must be countable. So class-0 sentences are countable.

What about class-1 sentences? Class-1 sentences are all class-0 sentences and all those sentences that can be formed out of class-0 sentences by applying a single connective once. Given this definition of class-1 sentences, a class-1 sentence must take one of the following forms:

- s
- $\neg s$
- $s \wedge t$
- $s \vee t$

- $s \supset t$

where s and t are class-0—i.e., atomic—sentences. So if we have a way of walking through each and every pair of class-0 sentences and listing the five class-1 sentences that can be formed out of each pair, we have a way of enumerating all class-1 sentences. Here is a procedure that will do the job:

Enumerating class-1 sentences Walk through the list of fractions and do the following: given the fraction n/m, list A_n, $\neg A_n$, $A_n \wedge A_m$, $A_n \vee A_m$, $A_n \supset A_m$ where any A_i is a class-0 sentence (i.e., an atomic sentence).

As we walk through the fractions, we will produce every class-0 sentence as well as any complex sentence that can be formed out of class-0 sentences by applying a single connective once. So this is a procedure for enumerating all class-1 sentences in groups of five.

What about class-2 sentences? We can enumerate them, too:

Enumerating class-2 sentences Walk through the list of fractions and do the following: given the fraction n/m, list S_n, $\neg S_n$, $S_n \wedge S_m$, $S_n \vee S_m$, $S_n \supset S_m$, where S_1, S_2, ...are all the class-1 sentences.

Since we know that class-1 sentences are countable, we know that there is a list of class-1 sentences S_1, S_2, ...that covers all class-1 sentences and that therefore this procedure will enumerate all class-2 sentences.

We could continue in this vein but there are infinitely many classes. We need a better way. And there is. The way we enumerate class-1 and class-2 sentences shows that for any n we can enumerate class-(n+1) sentences in the following way:

Enumerating class-(n+1) sentences Walk through the list of fractions and do the following: given the fraction i/j, list S_i, $\neg S_i$, $S_i \wedge S_j$, $S_i \vee S_j$, $S_i \supset S_j$, where S_1, S_2, ...are sentences of class-n.

If the sentence of class-n are countable, this procedure will enumerate all class-(n+1) sentences. That is, if class-n sentences are countable, so are class-(n+1) sentences. We can now see that for any n, class-n sentences are countable: class-0 sentences are countable. So class-1 sentences are countable. So class-2 sentences are countable. So class-3 sentences are countable. Etc.

But what about all the sentences of $\mathcal{L}$ taken together? That is, not just the sentences of a particular class-n, but all the infinitely many classes taken together. Can we enumerate them? That's easy, too:

Enumerating sentences of $\mathcal{L}$ Walk through the list of fractions and do the following: given the fraction n/m, list the m-th sentence of class-n sentences.

For any sentence of $\mathcal{L}$, it will eventually show up on the list of sentences generated in this way. Thus, the sentences of $\mathcal{L}$ are countable even when $\mathcal{L}$ has infinitely many atomic sentences. Put another way, you could assign a unique natural number as an ID to each and every sentence of $\mathcal{L}$ without worrying about exhausting the pool of ID numbers.

2.8.3 Things to note about the argument

Strictly speaking, we have shown more than that the sentences of $\mathcal{L}$ are countable. What we have shown is known as recursive enumerativity. Roughly, that means it is possible to write a computer program to generate a list of all the sentences of $\mathcal{L}$. Not everything that is countable can be enumerated by a computer program. For instance, consider a list of all the winning numbers of a lottery. That's countable, but if the lottery organizers know what they are doing, it is not possible to write a program that can generate the list of all winning numbers. The only way to list up all the winning numbers is by waiting for the drawings to take place (otherwise, you could get rich quick by writing a program that lists all the winning number, running it faster than the lottery numbers are drawn and buying the right tickets). There is something special and interesting about the fact that the sentences of $\mathcal{L}$ are recursively enumerable.

The procedure we used for enumerating the sentences of $\mathcal{L}$ will produce a highly redundant list because each class of sentences contains all members of the lower-numbered classes. If you don't like redundant lists, you can add the instruction 'if the sentence has been listed previously, don't list it'.

There are infinitely many ways of enumerating the sentences of $\mathcal{L}$. Usually sentences are enumerated for a purpose and the method of enumeration used is tailored for the purpose. The way we used here will be useful later for proving what is known as completeness of sentential logic (Section 4.8). A truly

momentous result known as Gödel's Incompleteness Theorem was proven using a different, very clever numbering scheme (That theorem concerns a formal language that is more complex than $\mathcal{L}$.)

On a more prosaic level, computers represent sentences as numbers (by assigning a number to each character and stringing them together). That's a way of assigning a unique ID to each distinct sentence. That works, and we don't have to worry about running out of numbers to represent sentences. Some numbering schemes are easy to reverse in the sense that given an ID number it is relatively easy to figure out which sentence it belongs to. Some schemes are harder to reverse. Digital encryption works by using schemes that are especially difficult to reverse without some extra information (the extra information is often called the decryption key).

What if $\mathcal{L}$ has only finitely many atomic sentences? In that case, each class of sentences has only finitely many sentences. But there still are infinitely many classes. You can use the same procedure for enumerating sentences of $\mathcal{L}$ by adding 'if there is no m-th sentence of class-n, skip to the next fraction'.

Suppose there are 100 students and each gets a unique ID number. If we add more students to the mix, we will need to get hold of *new* ID numbers to assign unique ID numbers to students in the enlarged group. Suppose there are infinitely many class-0 sentences. You can assign each sentence a unique ID so that each natural number is the ID of some sentence. That is, there are no numbers you can add to your pool of ID numbers. And yet, if you add class-1 sentences to the mix—there are infinitely many of them—you will be able to assign a unique ID to each and every sentence in that combined group of sentences by recycling the ID numbers you used for class-0 sentences. And you can do this even if you add infinitely many classes each of which has infinitely many sentences. That's a bit mind-boggling. You might suspect that there can be no such thing as having so many things you genuinely cannot give them unique IDs, but a very simple argument can show that that's not so. Let me close this chapter with a discussion of that.

2.9 Cantor's Diagonal Argument

Consider the way we produced the list of interpretations earlier. If there is only one atomic sentence, there are two interpretations. As we add an atomic sentence, the number of interpretations doubles. So if there are N sentences, there are 2^N interpretations. For instance, if there are 3 atomic sentences, there are $2^3 = 8$ interpretations. If there are 8 atomic sentences, there are $2^8 = 256$ interpretations. And so on. How many interpretations are there if there are infinitely many atomic sentences? Well, infinitely many. But is the number of interpretations countable? The answer is No. Here's an argument due to Georg Cantor (1891) known as the Diagonal Argument.

Take the way we have been listing interpretations when constructing truth tables. Imagine a list of interpretations constructed in that way for infinitely many atomic sentences by repeating the procedure infintely many times. We know exactly what such a list looks like. The first row has only T's. The second row starts with a single F and is followed by only T's. The third row starts with a T followed by an F and then only T's. The fourth row starts with two F's and then only T's. Etc. Starting from top of the list, the whole thing will look like Figure 2.2.

T	T	T	T	T	T	…
F	T	T	T	T	T	…
T	F	T	T	T	T	…
F	F	T	T	T	T	…
T	T	F	T	T	T	…
F	T	F	T	T	T	…
⋮	⋮	⋮	⋮	⋮	⋮	

Figure 2.2: interpretations for infinitely many atomic sentences

It may seem that the table must list all the interpretations. After all, we know exactly how to proceed as we increase the number of atomic sentences one by one. At no point will the procedure miss an interpretation, so how could the procedure miss an interpretation when extending it to infinitely many atomic sentences? And if the list of interpretations is complete, clearly the interpretations are countable: the first is on the first row, the second is on the second row, etc.

Matters, however, are not this simple. It is easy to construct an interpretation

that is not on the list. Here is how. An interpretation can be treated as a just a string of T's and F's. Let's construct new interpretation $\mathcal{N}$ in the following way:

- If the first letter of the first interpretation is a T, the first letter of $\mathcal{N}$ is F. If the first letter of the first interpretation is an F, the first letter of $\mathcal{N}$ is a T.
- If the second letter of the second interpretation is a T, the second letter of $\mathcal{N}$ is F. If the second letter of the second interpretation is an F, the second letter of $\mathcal{N}$ is a T.
- More generally, the n-th letter of $\mathcal{N}$ is a T iff. the n-th letter of the n-th interpretation is an F.

The following illustrates the method:

	(T)	T	T	T	T	T	…
	F	(T)	T	T	T	T	…
	T	F	(T)	T	T	T	…
	F	F	T	(T)	T	T	…
	T	T	F	T	(T)	T	…
	F	T	F	T	T	(T)	…
	⋮	⋮	⋮	⋮	⋮	⋮	
$\mathcal{N}$:	F	F	F	F	F	F	…

Figure 2.3: Diagonalization

It shows the top left corner of our list of interpretations. Each row above the horizontal line is an infinitely long interpretation. We construct the interpretation $\mathcal{N}$ below the horizontal line by taking each circled letter and 'flipping' it (T to F, F to T). $\mathcal{N}$ differs from the first interpretation because of the first letter, from the second interpretation because of the second letter, from the third interpretation because of the third letter, etc. $\mathcal{N}$ differs from each and every interpretation on the list. That means that $\mathcal{N}$ is not on the list. Thus, the list of interpretation is not a complete list. I hope you can see why the argument is called the *Diagonal* Argument.

The interpretation $\mathcal{N}$ we have found is the one that assigns F to all atomic sentences. And that cannot be on the list since given the way we construct the

list of interpretations, that particular interpretation must be the last one. But an infinite list cannot have a last line: that's what it is to be *infinte*—without end.

You might think we can handle that problem by simply adding the interpretation we have found to the list. Like this:

F	F	F	F	F	F	F	…
T	T	T	T	T	T	T	…
F	T	T	T	T	T	T	…
T	F	T	T	T	T	T	…
F	F	T	T	T	T	T	…
T	T	F	T	T	T	T	…
F	T	F	T	T	T	T	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	

Figure 2.4: list of interpretations with missing interpretation added

But this won't do. We can construct a interpretation that is not on this list:

	Ⓕ	F	F	F	F	F	F	…
	T	Ⓣ	T	T	T	T	T	…
	F	T	Ⓣ	T	T	T	T	…
	T	F	T	Ⓣ	T	T	T	…
	F	F	T	T	Ⓣ	T	T	…
	T	T	F	T	T	Ⓣ	T	…
	F	T	F	T	T	T	Ⓣ	…
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
$\mathcal{N}$:	T	F	F	F	F	F	F	…

Figure 2.5: Finding another missing interpretation

The point is completely general. It is not just that our particular way of generating the list of interpretations fails when dealing with infinitely many atomic sentences. Given any list of interpretations for infinitely many atomic sentences, we can find an interpretation that is not on the list. Here is an illustration to see that:

	Ⓣ	F	T	T	F	T	…
	F	Ⓣ	F	F	T	T	…
	T	T	Ⓕ	T	T	T	…
	F	F	F	Ⓣ	F	T	…
	F	F	F	T	Ⓕ	F	…
	T	T	T	T	F	Ⓣ	…
	⋮	⋮	⋮	⋮	⋮	⋮	
$\mathcal{N}$:	F	F	T	F	T	F	…

That means, the very idea of a complete list of interpretations when dealing with infinitely many atomic sentence is incoherent. That is what the Diagonal Argument shows. Given infinitely many atomic sentences, there are more interpretations than there are natural numbers. We say that the set of interpretations has higher *cardinality* than the natural numbers.

We now know that there really are more interpretations than there are atomic sentences—that's even true if the number of atomic sentences is infinite as we have just shown. What about the number of truth conditions?

A truth condition of a sentence specifies for each interpretation whether or not the sentence is true in that interpretation. That means that if $\mathtt{m}$ is the number of interpretations, the number of possible truth conditions is $2^{\mathtt{m}}$. For instance, if there are 8 atomic sentences, there are 256 interpretations and $2^{256} \approx 1.1 \times 10^{77}$ possible truth conditions. For comparison, our planet Earth is estimated to contain about 10^{50} atoms.[1] With just one more atomic sentence—9 of them—there are about $2^{512} \approx 1.3 \times 10^{154}$ possible truth conditions for a sentence. That vastly—and that's putting it very mildly—exceeds the estimated number of atoms in the observable universe which is around 10^{80} atoms.[2] This gives you a glimpse of the expressive capacities of natural languages: they can surely say more than what we can with just 9 atomic sentences in $\mathcal{L}$.

These considerations show that the number of truth conditions is larger than the number of interpretations in case the number of atomic sentences is finite. And we can prove that this is so even if there are infinitely many atomic sentences.

The proof proceeds via a slight reformulation of the Diagonal Argument. Notice that the question whether or not there are countably many interpreta-

[1] https://www.fnal.gov/pub/science/inquiring/questions/atoms.html
[2] https://en.wikipedia.org/wiki/Observable_universe

tions is the same as the question whether or not it is possible to pair atomic sentences with interpretations without residue on either side. An interpretation can be understood as a subset of atomic sentences: each interpretation contains those atomic sentences that are true in that interpretation. So the question is whether or not the set of atomic sentences can be paired, without residue on either side, with sets of atomic sentences—these latter are the subsets of the set of atomic sentences. Take any set $\mathcal{S}$ and consider a pairing of its members with subsets of $\mathcal{S}$. Could such a pairing be one-to-one without residue? The answer is no. We can construct a new subset $\mathcal{N}$ in the following way: for any member e of $\mathcal{S}$, e is in $\mathcal{N}$ iff. e is not in the subset of $\mathcal{S}$ paired with it. $\mathcal{N}$ is not paired with any member of $\mathcal{S}$ since for any member e of $\mathcal{S}$, $\mathcal{N}$ differs from the subset of $\mathcal{S}$ paired with e. This shows that the cardinality of the set of all the subsets of $\mathcal{S}$—a.k.a. the *power set* of $\mathcal{S}$—must be larger than the cardinality of $\mathcal{S}$. This is known as Cantor's Theorem.

The argument just given can be given for any set and its power set: thus, as we have already seen, the set of interpretations has higher cardinality than the set of all atomic sentences. Now, a truth condition is a set of interpretations: the set of all the interpretations in which a sentence is true. So the cardinality of truth conditions is higher than the cardinality of interpretations. And the set of all the subsets of truth conditions has even higher cardinality, and so on. This opens up a realm of mathematical entities that David Hilbert called Cantor's paradise (1926).

Chapter 3

Proofs: Natural Sequent Calculus

3.1 Arguments

In the previous chapter we defined a formal language. Consider native speakers of such a formal language $\mathcal{L}$. They can say many things: not just the atomic sentences, but an infinite variety of compound sentences using the logical connectives $\neg$, $\wedge$, $\vee$, and $\supset$. Not only are they capable of formulating an infinite variety of sentences, they are also capable of saying sentences with a large variety of different truth conditions, i.e., meanings. Just how large that variety is depends on the number of atomic sentences, of course.

But we do more than just say things using language. One very important use of language is the presentation of *arguments*. Consider:

> **Example 3.1.1.** More investment [in indoor ventilation] would be money well spent. Better indoor air boosts academic performance—maths and reading scores go up. Office-workers benefit, too. Researchers have found the cognitive scores of people in well-ventilated offices are 61% higher than those of workers in conventional office set-ups. (from *The Economist*, May 29, 2021)

This little snippet consists of four sentences. But it is not simply an enumeration of four facts. Rather, it is trying to persuade the reader that investment in indoor ventilation would be money well spent. And it attempts that not by bluntly asserting the claim but by providing reasons in support of the claim that investing in indoor ventilation makes sense. In short, it is an argument.

The claim that an argument is meant to support is called the *conclusion*. A claim adduced in support of the conclusion is called a *premise*. Arguments often have multiple premises. In the above example, the conclusion is that investment in indoor ventilation is money well spent. And there are two stated premises: (a) better indoor air boosts academic performance; and (b) researchers have found the cognitive scores of people in well-ventilated offices are 61% higher than those of workers in conventional office set-ups.

Our main topic of interest in this chapter is arguments. We will be looking at highly formalized arguments, but before doing that let me discuss natural language arguments a bit more. It will help understanding the virtues of the formalizations we will be studying.

Consider:

> **Example 3.1.2.** From April, British prisons will have to screen all

> inmates who have experienced domestic violence for brain injuries. Such screening should be extended to all prisoners. It would enable staff to identify those whose brains have been damaged and offer them appropriate support. (from *The Economist*, March 27, 2021)

The conclusion of this argument is clear enough: all prisoners should be screened for brain injuries. But how is that conclusion supported? Apart from the conclusion, the snippet states two facts:

(a) From April British prisons have to screen all inmates who have experienced domestic violence for brain injuries.

(b) Screening for brain injuries enables staff to identify those whose brains have been damaged and offer them appropriate support.

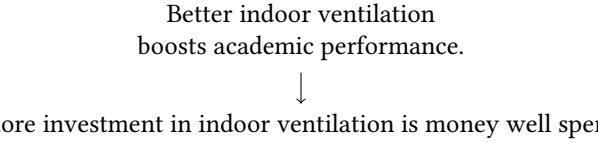
Does (a) support the conclusion? Upon a little reflection, it seems not: even if British prisons did not screen any inmates, that would not weaken the case for screening them for brain injuries. So (a) is in fact not a premise. So the argument as stated has only one premise: (b). But does (b) support the conclusion? Taking (b) to support the conclusion requires accepting that even criminals deserve support for dealing with the effects of brain injuries. You may find that obvious, but others find that highly controversial. The point: the snippet above contains a claim that is no part of the argument, and the argument depends on an unstated premise about what criminals deserve. And, if you think about it, the first example (Example 3.1.1) also makes some unstated assumptions about what is worthy of spending money on. This is a common feature of arguments stated in natural languages. We need to be careful to make sure we do not miss hidden premises, and be careful not to mistake some extraneous information as part of the argument.

3.2 Diagrammatic Representations of Arguments in Tree Form

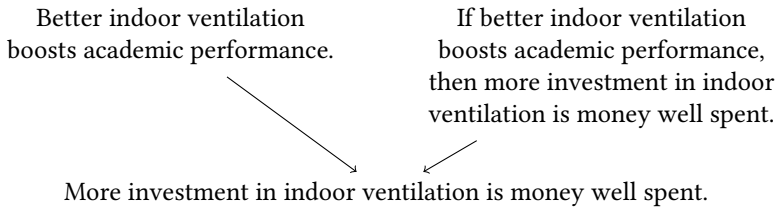
An argument presents certain connections among thoughts. In particular, they present how some thoughts support other thoughts. You probably are familiar with things like mind maps as ways of presenting connections between ideas. Arguments can be presented in a similar fashion as diagrams. For instance, Consider a simplified version of the argument in Example 3.1.1:

> **Example 3.2.1.** More investment [in indoor ventilation] would be money well spent. Better indoor air boosts academic performance.

We can represent this in the following way:



The arrow represents the direction of support, so the idea expressed at the top of the diagram supports the idea expressed at the bottom. But, as already pointed out, it seems that there is another thought that is playing a role in supporting the conclusion. We can capture that:



It is the two thoughts at the top taken together that support the conclusion at the bottom. The above is a very common pattern of argument known as *modus ponens*. We can make the pattern clear by replacing the sentences with letters.

Let A stand for ⟨Better indoor ventilation boosts academic performance⟩ and let B stand for ⟨more investment is money well spent⟩. Then the above can be represented as:

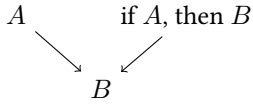


Figure 3.1: Modus Ponens

The sentences at the top are the premises, the sentence at the bottom the conclusion. If you think about the meaning of the conditional, it is clear that if the two premises are true, the conclusion must be, too. So the premises do provide good support for the conclusion, and it does not matter what A and B stand for.

We can also have more complex arguments. For instance, if you are asked why we should accept that better indoor ventilation boosts academic performance, we might answer that test scores go up when ventilation is improved, and that if scores go up, then that shows that better indoor ventilation boosts academic performance. We could represent the augmented argument by the following diagram which has modus ponens occurring twice—once to get to A, and then to get to B:

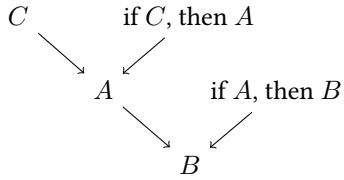


Figure 3.2: Complex Argument

More complicated arguments will be represented by larger diagrams. Arguments might be thought of as having structures akin to river systems: smaller streams flow together to form ever larger systems until they reach the conclusion. Investigating various structures of arguments reveals a few very common patterns of 'confluences'. The above patterns which combines a condi-

tional and the antecedent of the conditional is among the most common patterns. Let me discuss some more common patterns.

Consider the following piece of reasoning:

> **Example 3.2.2.** The Central Bank will stop raising interest rates only if inflation has been brought under control. But inflation has not been brought under control. So the Central Bank will not stop raising interest rates.

This has the following structure known as *modus tollens* (recall that ⟨if A, then P⟩ is another way of saying ⟨A only if B⟩):

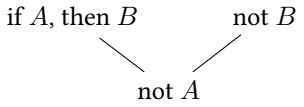


Figure 3.3: Modus Tollens

We omit the arrow heads: in such argument diagrams, the support is always from top to bottom.

Here is an example of another common pattern. Socrates—often regarded as the founder of Western philosophy—argued that death is a good thing in the following way: death is one of two things; either it is like an eternal deep sleep which is a good thing; or death means moving to another world where you can spend an eternity talking to such interesting people like Homer and the ancient Greek heroes, which is a good thing; so either way death is a good thing; thus, death is a good thing.

This is known as an *argument by cases*. Here is a way of representing this:

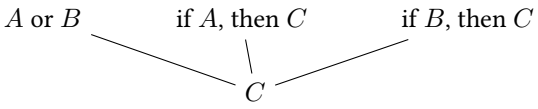


Figure 3.4: Argument By Cases

Here we have a confluence of three claims.

Here is one more example of a common pattern. Consider a well-known argument that you cannot go back in time and kill your biological mother before you were even conceived:

> **Example 3.2.3.** You plan to go back in time and kill your mother before you were even conceived. Suppose your plan succeeds. In that case, your mother does not conceive you, and you do not exist. On the other hand, for your plan to succeed, you must execute the plan, and that requires you to exist. So if your plan succeeds, you do and do not exist, which is a contradiction. Thus, your plan cannot succeed.

This is an instance of what is known as *reductio ad absurdum* or argument by contradiction. This pattern can be represented as:

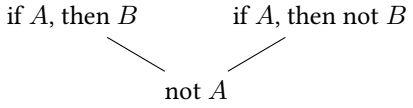


Figure 3.5: Reductio ad Absurdum

3.3 Arguments in Standard Form

Diagrammatic representations of arguments can be very helpful for understanding the structure of larger arguments, but they take a lot of space and are difficult to produce without the help of technology. A far more common way of representing an argument is in the form of a list of sentences. A simple list of sentences is a one-dimensional structure and cannot give us all the information that the two-dimensional diagrams in the previous section can represent. To make up for this shortcoming, we will add *annotations* to each line that gives us extra information about how the sentences connect. For instance, Modus Ponens is represented as:

1. If A, then B . . . . . premise
2. A . . . . . premise
3. B . . . . . 1,2, MP

The annotations at the end of each line give us enough information to reconstruct the diagrams of the previous section. The keyword 'premise' in the annotations of lines 1 and 2 tells us that they are not represented as supported by further nodes up the diagram. The '1,2' in the annotation of line 3 tells us that the sentence on line 3 is supported by the sentences on lines 1 and 2. And this is enough to enable us to reconstruct the argument tree:

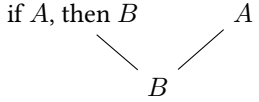


Figure 3.6: Modus Ponens

The 'MP' in the annotation of line 3. names the pattern Modus Ponens. As you can see, we only need the references to the supporting lines in the annotation to reconstruct the argument diagram, but it helps to explicitly name the pattern as it helps with checking whether the conclusion really is supported by the premises.

From here on, the normal way of presenting arguments will be in such list form. While I will occasionally display an argument in tree form when it helps

making the structure of an argument more vivid, you do not have to produce such diagrams yourself.

Apart from making sure that the list contains enough information to reconstruct the tree, we also require that an argument in *standardized form* be such that a given line cannot be supported by lines below it. So the direction of support is strictly from top to bottom. That helps with comprehension of the argument presented.

When an argument is such that the conclusion cannot fail to be true if all the premises are true, it is said to be *valid*. One of the important tasks of logic is figuring out which argument patterns are valid.

Let me go through the patterns of arguments discussed in the previous section. They are all known to be valid. You can check that they are valid by checking truth tables: could all the premises be true without the conclusion being true?

3.3.1 Modus Ponens (MP)

I already have shown how an instance of the use of modus ponens works:

1. If A, then B . . . . . premise
2. A . . . . . premise
3. B . . . . . 1,2, MP

Any argument with a premise P and conclusion C can be turned into a logically valid argument by adding the premise ⟨if P, then C⟩. For instance, suppose someone reasons, "Donald is rich; therefore, he is fit to be the president of a country." We can turn this into a logically valid argument as follows:

Argument 1.

1. If Donald is rich, then Donald is fit to be the president of a country. . . . . premise
2. Donald is rich. . . . . premise
3. Donald is fit to be the president of a country. . . . . 1,2, MP

While this is a valid argument—*if* both premises are true, then the conclusion is also true—this need not give us a particularly strong reason to accept the

conclusion: putting it cautiously, some people might think that recent history decisively shows that the first premise is false. A benefit of presenting arguments in standardized form is that it enables us to see what all the needed premises are which in turn allows us to see which premises might be suspect.

3.3.2 Modus Tollens (MT)

Here is Modus Tollens in standard form:

1. If A, then B. . . . . premise
2. Not B. . . . . premise
3. Not A. . . . . 1,2, MT

Here is an example using 3.2.2:

Argument 2.

1. If the Central Bank will stop raising interests rates, then inflation has been brought under control. . . . . premise
2. Inflation has not been brought under control. . . . . premise
3. The Central Bank will not stop raising interest rates. . . . . 1, 2, MT

3.3.3 Argument By Cases (AC)

Here is the argument by cases in standard form:

1. A or B . . . . . premise
2. If A, then C . . . . . premise
3. If B, then C . . . . . premise
4. C . . . . . 1,2,3, AC

If all premises are true, then at least one of A and B is true (by premise 1). If A is true, then C is true (premise 2). If B is true, then C is true (premise 3). So C is true given all three premises are true. Thus, this argument form is valid.

Of course, just because such an argument is valid does not mean that the argument is a good one. For instance, you might wonder whether Socrates is right that death must be one of the two things he lists; and even he is right about that, is it so obvious that death would be a good thing on both options—a day talking to Homer might be super interesting, but a whole eternity talking to him?

3.3.4 Reductio ad Absurdum (RAA)

Here is Reductio ad Absurdum:

1. If A, then B . . . . . premise
2. If A, then not B . . . . . premise
3. Not A . . . . . 1,2, RAA

In the example about time travel we discussed earlier, let A be 'your plan succeeds' and B be 'you exist' which gives us:

Argument 3.

1. If your plan succeeds, then you exist. . . . . premise
2. If your plan succeeds, then you are do not exist. . . . . premise
3. Your plan does not succeed. . . . . 1,2, RAA

3.3.5 Extracting arguments from natural language texts

Extracting arguments from natural language texts and putting them in standardized form takes some practice. Let me start with an example that illustrate some of the tricky aspects.

Julius Caesar was assassinated by a group of Roman senators led by Brutus. The conspirators accused Caesar of harboring ambitions of becoming the king of Rome (Rome was a republic at the time). In Shakespeare's fictionalized version of the events in the play *Julius Caesar*, Marc Antony—a prominent protégé of Caesar—gives a speech at Caesar's funeral that turns the crowd against Brutus. Here is a snippet from that speech:

> You all did see that on the Lupercal
> I thrice presented him a kingly crown,
> Which he did thrice refuse: was this ambition?
> Yet Brutus says he was ambitious;
> And, sure, he is an honourable man.

Notice that Marc Antony's message is that Caesar was *not* ambitious, and that Brutus is *not* honourable, neither of which is said explicitly: the former is indicated by a rhetorical question, and the latter is indicated by an ironic assertion of the opposite. We can represent the case Marc Antony is making as two arguments. Here is the first part:

1. If Caesar refused a kingly crown, then he was not ambitious. . premise
2. Caesar refused a kingly crown. . . . . premise
3. Caesar was not ambitious. . . . . 1,2, MP

And here is the second part:

1. If Brutus says Caesar was ambitious, then Brutus is not honourable. . . . . premise
2. Brutus says that Caesar was ambitious. . . . . premise
3. Brutus is not honourable. . . . . 1,2, MP

These arguments use Modus Ponens. We could also represent the arguments using Modus Tollens:

1. If Caesar was ambitious, then he did not refuse a kingly crown. . . . . premise
2. Caesar refused a kingly crown. . . . . premise
3. Caesar was not ambitious. . . . . 1,2, MT

And

1. If Brutus is honorable, then Brutus does not say Caesar was ambitious. . . . . premise
2. Brutus says that Caesar was ambitious. . . . . premise
3. Brutus is not honourable. . . . . 1,2, MT

It is normal for there to be multiple ways of representing an argument stated in natural language. Which standardized version you choose is mostly a matter of taste and convenience.

You might feel that the standardized arguments using Modus Tollens don't sound like good English. For instance, it might sound more natural to say 'if Caesar was ambitious, he wouldn't have refused a kingly crown', instead of the formulation given above. We will ignore such niceties.

3.4 Shortcomings of Standardized Form

Presenting arguments in standardized form helps a great deal in making sure that an argument is valid. But it still has some weaknesses—or, rather, the diagrammatic representation we had earlier has weaknesses. To understand the weaknesses, consider the example we used to illustrate Reductio ad Absurdum earlier about your plan to kill your mother before you were even conceived. One way of putting the point is that if your plan succeeds, then you are unable to execute the plan. We can try to present the argument for this in standard form:

1. Your plan succeeds. . . . . assumption
2. If your plan succeeds, then you don't exist. . . . . premise
3. You don't exist. 1,2,MP
4. If you don't exist, then you are unable to execute the plan. . . . . premise
5. You are unable to execute the plan. 3,4,MP
6. If your plan succeeds, then you are unable to execute the plan. . . . . ??

Notice that line 1 is explained as an *assumption* rather than a *premise*. An assumption is just that. Unlike a premise that you can call into question by asking what reasons there are for accepting it, an assumption requires no such reasons to accept it. We can assume whatever we please to see what follows from that assumption. Sometimes, such assumptions bear useful fruit (often not). Many formal presentations of arguments do not distinguish between premises and assumptions and instead treat all premises as assumptions. We will not do that here. Assumptions are claims that we are free to introduce and reject. For instance, in the argument just discussed, the assumption that your plan succeeds will ultimately be rejected. Premises are not like that. They are given as true and are, in the context of the argument, not up for rejection. So when we mark something as a premise, we mark it as a claim to be taken for granted—and in doing so, we are also vouching that there is a good reason to take the premise to be true; that reason might be explicit in the surrounding context, but even if it is not, it is something that needs to be produced upon request.

Now to the more important point for our current purposes. What should go into the annotation for line 6? It is tempting to think of the structure of the argument as given by the following tree (line number are kept):

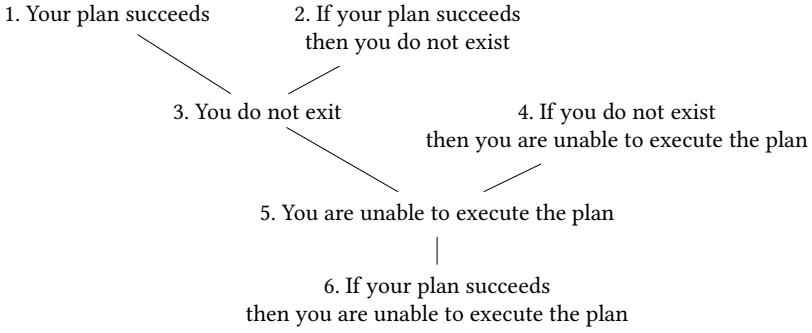


Figure 3.7: Attempt 1

But this is somewhat misleading because it suggests that it does not matter how we get to ⟨you are unable to execute the plan⟩ even though it clearly does matter: the antecedent of the conditional in the conclusion is taken from the assumption made at the start which we need to get to 5. What we are missing is a clear way of indicating how a given node might depend on how previous nodes are supported.

Let's think about this a little more. Think about what the tree above is telling us up to ⟨you are unable to execute the plan⟩. The branch on the left tells us that ⟨you do not exist⟩ (the node numbered 3) follows from, or is supported by, ⟨ your plan succeeds⟩ and ⟨if your plan succeeds, then you do not exist⟩. Let's put this as:

- Given ⟨your plan succeeds⟩ and ⟨if your plan succeeds, then you do not exist⟩: ⟨you do not exist⟩.

The tree also tells us (inspecting the node numbered 5):

- Given ⟨you do not exist⟩ and ⟨if you do not exist, then you are unable to execute the plan⟩: ⟨you are unable to execute the plan⟩.

Comparing these two point, we get:

- Given ⟨your plan succeeds⟩ and ⟨if your plan succeeds, then you do not exist⟩ and ⟨if you do not exist, then you are unable to execute the plan⟩: ⟨you are unable to execute the plan⟩.

And from this we can conclude:

- Given ⟨if your plan succeeds, then you do not exist⟩ and ⟨if you do not exist, then you are unable to execute the plan⟩: ⟨if your plan succeeds, then you are unable to execute the plan⟩.

Let S be ⟨your plan succeeds⟩, E be ⟨you exist⟩, and P be ⟨you are able to execute the plan⟩. We can get a better representation of what is going on in our reasoning by the following tree:

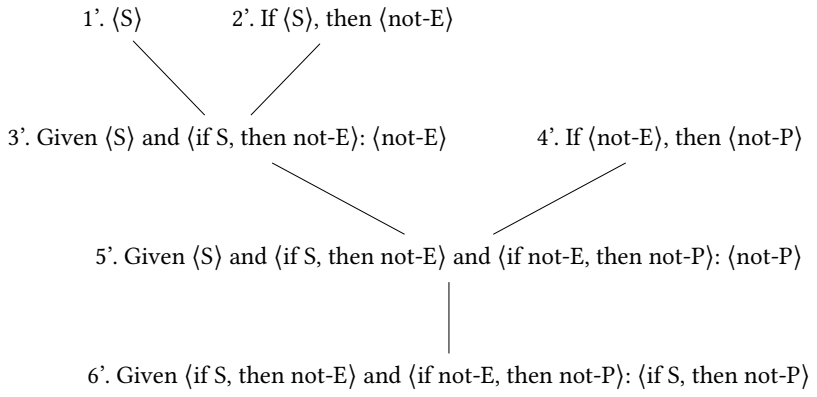


Figure 3.8: Attempt 2

Here the nodes below the premises/assumptions indicate explicitly what is needed to support what. You might think this is redundant since the lines connecting the nodes already indicate what supports what. But notice that this fixes the earlier problem of not making clear that 6 depends on line 5's following in the way it does from the premises: if 5' said ⟨given D, it follows

that $not-P\rangle$, 6' would not follow. So the extra information we are adding to the nodes is not always redundant.

In list form, the argument can be represented as:

1'. ⟨S⟩ . . . . . assumption

2'. If ⟨S⟩, then ⟨not-E⟩ . . . . . premise

3'. Given ⟨S⟩ and ⟨if S, then not-E⟩: ⟨not-E⟩ 1,2

4'. If ⟨not-E⟩, then ⟨not-P⟩ . . . . . premise

5'. Given ⟨S⟩ and ⟨if S, then not-E⟩ and ⟨if not-E, then not-P⟩: ⟨not-P⟩ 3,4

6'. Given ⟨if S, then not-E⟩ and ⟨if not-E, then not-P⟩: ⟨if S, then not-P⟩ . 5

Notice that this does not have any inference patterns in the annotations. That is because the nodes look differently from before so that we would have to reformulate our inference patterns to make them usable with this style of regimenting arguments. But we will not bother with that as we will move on to a much more highly formalized way of presenting arguments.

3.5 Keeping Track of Support Relations

From here on, we will represent sentences using our formal language $\mathcal{L}$ developed earlier, unless there is reason to revert to natural language sentences. This will enable us to be concise, precise, and some things we want to show about arguments will be easier to handle.

Toward the end of the last section, we represented arguments using lines like 'given ... : ...'. We will shorten this using a device known as a *sequent*. A sequent looks thus:

$$s_1, s_2, ..., s_n \vdash c$$

The $\vdash$ symbol is known as the *turnstile*. A sequent consists of a turnstile symbol flanked by a list of sentences on the left and a single sentence on the right hand side. You can read $s_1, s_2, ..., s_n \vdash c$ as 'given $s_1, s_2, ..., s_n$, c follows' or '$s_1, s_2, ..., s_n$ supports c'. We will call the left side the *datum* (the given), and the right side the *succedent*. So a sequent tells us that its datum supports the succedent. We will have to sharpen our understanding of a sequent in due course, but this suffices for now. Let me just note that the datum is allowed to be an empty list, and that there must be a single sentence on the succedent side.

You might have noticed that the premises and assumption in the last section did not take the 'given... : ...' form. In our formal presentation of arguments, we want all lines to take a uniform form. That is, we want each line to be a sequent. How can we do that? Let's think about premises and assumptions one by one.

A premise is a claim that is taken to have some real reason to accept it. So in principle, we can say what it is that supports a premise. But in practice, since we cannot say everything in a finite amount of time and space, we simply ask the audience to accept a claim as in fact supported even if the grounds are unspecified. We can mimic this by using place holders. We will use upper case Greek letters to stand for a (possibly empty) list of sentences. So

$$\Gamma \vdash c$$

says that given all the sentences in Γ, c follows (Γ is upper case Gamma). E.g., we can represent the argument in Example 3.2.1 like this:

1. Γ $\vdash A$ ..... premise
2. Δ $\vdash A \supset B$ ..... premise
3. Γ, Δ $\vdash B$ 1,2

(Δ is Greek uppercase delta.)

What does this representation of the argument say? It tells us that given some consideration Γ, A follows; given some consideration Δ, $A \supset B$ follows. We are not told what those considerations are, so the Greek letters are just placeholders to be filled in upon request. But those considerations support A and $A \supset B$. And the last line tells us that the considerations supporting A and $A \supset B$ taken together support B. Notice that it really must be Γ and Δ taken together that supports B. Γ alone wouldn't support B because the considerations in favor of A need not be considerations in favor of $A \supset B$; and the considerations in favor of $A \supset B$ need not be considerations in favor of A. We need both the considerations in favor of A and the considerations in favor of $A \supset B$ to get support for B and that is what line 3 tells us. (The spacing around the turnstile and the small font size on the datum side is for visual effect only. You don't need to replicate that when you write them by hand.)

What about assumptions? How should we represent an assumption in sequent form? An assumption is not supported by any evidence. You might therefore be tempted to use a sequent with an empty datum since wouldn't that say that nothing supports the succedent? But that sort of sequent will be reserved for something else (you will see the wisdom of this as we progress). We will represent an assumption using a sequent of the following form:

$$s \vdash s$$

This says that given s, s follows. And that is true. But it also fails to give us any reason to accept s since no claim can support itself. And this is exactly what an assumption is like: an assumption gives us no reason to accept that the assumed sentence is actually true. Now that we have a way of representing assumptions, we can represent the argument we discussed at length near the end of the previous section:

4. S $\vdash S$ ..... assumption
5. Γ $\vdash S \supset \neg E$ ..... premise

6. $\Gamma, S \quad \vdash \neg E \quad \ldots\ldots 4,5$
7. $\Delta \quad \vdash \neg E \supset \neg P \quad \ldots\ldots$ premise
8. $\Gamma, \Delta, S \quad \vdash \neg P \quad \ldots\ldots 6,7$
9. $\Gamma, \Delta \quad \vdash S \supset \neg P \quad \ldots\ldots 8$

Notice that S does not appear on the datum side of line 9. And that is important: whatever considerations there are in support of the succedents of lines 5 and 7, those also suffice to support the succedent of line 9: $S \supset \neg P$. Whether or not an assumption is needed to support the succedent of a given sequent matters. For instance, consider the following argument in the initial standard form we used earlier:

10. If the Moon is made of cheese, then the Moon is edible ..... premise
11. The Moon is made of cheese ..... assumption
12. The Moon is edible 10,11

Just a little bit of reflection shows that no reason has been given to accept that the Moon is edible: line 11 is labeled an assumption and that means no pretense is made that there is any reason to accept that the Moon is made of cheese, but we would need that to accept the conclusion as true. Merely inspecting line 12 does not reveal that. In sequent form, however, the flaw is very apparent, since the argument above has the form:

13. $\Gamma \quad \vdash A \supset B \quad \ldots\ldots$ premise
14. $A \quad \vdash A \quad \ldots\ldots$ assumption
15. $\Gamma, A \quad \vdash B \quad \ldots\ldots 13,14$

A shows up in the datum of the concluding line. Because we use Greek upper case letters for what supports premises, if anything other than Greek upper case letters appear in the datum of the conclusion, we know that the conclusion does not follow from the premises alone. So in line 15, the succedent B does not follow from what supports the premise on line 13. (Some Greek upper case letters case look like Roman upper case letters. Don't use them as they are likely to cause confusion.)

You might think that whenever an argument makes use of an assumption, the conclusion will be flawed in that it has no actual support. That is not so: look at the argument from line 4 to 9 above. It makes an assumption on line 4, but the datum of the assumption does not show up in the datum of line 9. All that you need to support the succedent of line 9 are whatever considerations support the succedents of the premises (lines 5 and 7). Logicians say that the assumption on line 4 has been *discharged* (like soldiers being discharged—their services are no longer required). The sequent notation of arguments can make explicit this feature of the use of assumptions.

There is one important point that I should mention here. The turnstile symbol $\vdash$ is not a part of $\mathcal{L}$. And a sequent is also not a sentence of $\mathcal{L}$. It is, in fact, a sentence of our meta-language that expresses a fact about some sentences of $\mathcal{L}$, viz. that some sentence follows from some other sentences.

3.6 Inference Rules

An argument is a series of sequents. When we draw a conclusion in an argument, what we do is start with a series of sequents and then add another sequent on the basis of some sequents already present. Clearly, not everything goes. Some ways of extending the series of sequents are acceptable, others not. That is, some arguments work, others don't. But so far, we do not have an easy way of checking whether a proposed argument actually works. To facilitate checking arguments, we will introduce a number of *inference rules* that govern the construction of formal arguments. Only arguments that are in accordance with these inference rules are acceptable.

For terminological clarity, we will call formalized arguments *derivations*. And we stipulate that a derivation consists of a finite number of sequents—so if Socrates were to have an argument with Homer going on for all eternity, the argument cannot be turned into a derivation. A derivation *derives* sequents from one or more preceding sequents. Each step in the derivation must be licensed by one or another inference rule. When we make a step in accordance with an inference rule, we will say that we *infer* a sequent from one or more preceding sequents (there is a special case where a sequent is 'inferred' from no preceding sequent; we will discuss that soon).

Keep in mind that only arguments formalized in accordance with our proof system are derivations. We will encounter lots of arguments stated in English—like all the arguments about our system of logic that we must give in the meta-language. Those are not called derivations. They are simply arguments.

In looking for inference rules, we want rules that are completely obvious. No one who understands what a given rule says should be able to cogently doubt that reasoning in accordance with the rule is indeed rationally permissible. But we will not be formulating all inference rules that are obvious: when we construct derivations, we want to minimize the appeal to obviousness.

Apart from a few trivial rules, we will see a pair of inference rules for each of the connectives $\neg$, $\wedge$, $\vee$, $\supset$: an *introduction* rule and an *elimination* rule. The intuitive idea behind an introduction rule for a given connective is that it allows you to infer from some premises to a conclusion that has that connective as its main connective—it *introduces* the connective into the train of thought. An elimination rule allows you to infer from a premise that has the connective as the main connective to a conclusion that does not, often in connection with

other premises (though there are some complications which will see in a bit).

We have been speaking of inference *rules*. Let me say a few words about the notion of rules at work.

Sometimes, a rule says you must do something under certain circumstances. For instance, in many jurisdictions you must pay income tax if you have income. It's not that you have the option to pay income taxes but you don't have to (wouldn't that be nice...). This is certainly one kind of rule: they require you to do something if a certain condition is met. Call these *rules of requirement*.

But not all rules are rules of requirement. For instance, in many jurisdictions there are rules about who may run for office. Typically, if you were born to the right parents and/or in the right place and are old enough, you are eligible to run for public offices. This does not mean that if you meet the eligibility criteria you are required to run for office. Rather, if you meet those criteria, you are permitted to run for office. This is another kind of rule: they permit you to do something if certain conditions are met. Call these *rules of permission*.

A rule of permission requires a general prohibition in the background. A rule of permission gives you permission to do something that you otherwise are not allowed to do. For instance, you are not allowed to run for office unless you meet certain conditions. So you could think of rules of permission as rules that create exemptions to a general prohibition.

Rules of inference are rules of permission. They are rules that govern how you may extend a series of sequents. There is a general rule that says: you are not allowed to extend a series of sequents in any way. If we only had this, we could not construct derivations. The rules of inference create exemptions. They say that if a given series of sequents meets certain conditions, you *may* extend it in a certain way. Whether you will extend it in that way is up to you. But unless one of the inference rules says that you may do it, you are not allowed to do it. This imposes serious constraints on how we can construct derivations, but this will also enable us to formulate derivations that are much tighter than ordinary arguments.

The combination of our formal language and the inference rules is known as the *proof system* of sentential logic.

> The proof system I will be presenting here was developed by Gerhard Gentzen (1936). It is in fact the same system as the one E.J. Lemmon (1978) uses except that in his style the sequents come in a bit of disguise.

Lemmon and systems influenced by him (e.g., Allen and Hand (2001)) present the datum side of sequents to the left of the line number, and instead of sentences in the datum, the line numbers where a sentence was first introduced is used. So the argument from 1 to 3 in this section looks in Lemmon style as follows:

1	(1)	A	...premise
2	(2)	$A \supset B$	...premise
1,2	(3)	B	 1,2

The numbers in the middle in parentheses are the line numbers. The '1' to the left of line numbers refers to whatever supports the succedent of line 1, and the '2' refers to whatever supports the succedent of line 2. You can turn the above Lemmon style presentation into our style by first replacing 1 with A and 2 with $A \supset B$, then moving the line numbers all the way to the left, and finally inserting the turnstile in the gap created. You will see that the 'premises' take the form of assumptions.

Lemmon's style has the advantage of looking pretty much like the standardized presentation of arguments if you squint a little and ignore what's to the left of the line numbers—but you could say the same about our system: squint a little harder and ignore what's between the line number and the turnstile. Lemmon's style of presenting derivations has the disadvantage of not being able to distinguish premises and assumptions and, more importantly, some of the features of the proof system that we will be investigating are harder to deal with. Once you get over the novelty of writing sequents, the Lemmon style notation has no real advantages so we will go with sequents from the start.

Gentzen introduced two more very well-known proof systems in his (1935a; 1935b). One of those also uses 'sequents' but the definition of a sequent in that earlier work is different from the one used in Gentzen's (1936). I follow the latter version of 'sequent calculus' here.

3.7 Preliminaries: Rewriting Sequents

Our inference rules will be formulated using sequents. There are some features of sequents that are very useful for applying those rules. Let me briefly go over them before launching into the inference rules proper.

3.7.1 Reordering lists

The datum of a sequent is simply a list of sentences. The order in which the sentences are listed is irrelevant. That is obvious if you think about what a sequent means. If the considerations A and B support the claim C, then that fact does not depend on the order in which the considerations are stated.[1]

So you may rewrite a sequent by reordering the datum side as you see fit. Perhaps, you want them to be in alphabetical order, or roundish looking letters before the squarish ones, etc. For instance, the following is fine:

1. $\Gamma, C \quad \vdash E \quad \ldots\ldots$ some conclusion
2. $C, \Gamma \quad \vdash E \quad \ldots\ldots 1$

3.7.2 Redundant lists

Sometimes, you will find yourself with a sequent that has duplicate items on the datum side. E.g., $s, s \vdash t$. Stating the consideration s twice does not affect the support that s provides for t.[2] So we can simply get rid of the duplicate:

3. $S, S \quad \vdash T \quad \ldots\ldots$ some conclusion
4. $S \quad \vdash T \quad \ldots\ldots 3$

[1] Humans often are swayed by the order in which considerations are presented. That is why, for instance, lawyers pay attention to the order in which witnesses are called. But this is simply a symptom of human irrationality. If the defendant has a motive to commit the crime they are accused of, that fact supports the claim the defendant is guilty, and it does not matter whether that consideration is stated at the start of the trial, near the end, or buried among myriad other facts.

[2] Joseph Goebbels is often credited with the observation that if you repeat a lie often enough people believe it. This is another instance of human irrationality. Our proof system is designed to guard against such failures. By the way, there apparently is no evidence that Goebbels ever made this particular observation about the effect of repetition. But as the saying goes, if you repeat something often enough...

3.7.3 Summary

You may use the following principles for rewriting the datum side of sequents:

a. You may reorder items within the datum as you see fit.

b. You may delete duplicate items within the datum.

When you rewrite a sequent using these rewrite rules, the annotation should just say which line you rewrote—see the examples above. As you will see, as we get used to constructing derivations, these rewrites are often done silently in combination with the application of other inference rules.

3.8 Conditional Elimination and Assumption Introduction

3.8.1 Conditional Elimination

Let us start with Modus Ponens. The version in our formal proof system is called *Conditional Elimination* rule. It says that you may do the following:

Conditional Elimination (⊃E) From $\Lambda_1 \vdash s_1 \supset s_2$ and $\Lambda_2 \vdash s_1$, infer $\Lambda_1, \Lambda_2 \vdash s_2$.

Λ is Greek upper case Lambda. This rule requires the presence of two sequents: one must be a sequent whose succedent has the conditional as its main connective; and the other's succedent must be the antecedent of the conditional. Now, if a series of sequents has sequents of the requisite forms present, you may extend the series by a sequent with a succedent which is the consequent of the conditional, and whose datum is the union of the two datums of the sequents you start with.

We can now represent the argument employing Modus Ponens in Example 3.2.1 by the following derivation :

1. $\Gamma \quad \vdash A \quad \ldots\ldots$ premise
2. $\Delta \quad \vdash A \supset B \quad \ldots\ldots$ premise
3. $\Gamma, \Delta \quad \vdash B \quad \ldots\ldots$ 1,2,⊃E

Γ is Λ_2 and Δ is Λ_1, A is s_1 and B is s_2. The annotation of line 3 says that we got to it by applying Conditional Elimination to lines 1 and 2. You might notice that the order of the sequents numbered 1 and 2 is different from the way in which they are listed in the statement of ⊃E. That is ok. Because what the rule says is that the following derivation tree is permissible:

$$\begin{array}{ccc} \Lambda_1 \vdash s_1 \supset s_2 & & \Lambda_2 \vdash s_1 \\ & \diagdown \quad \diagup & \\ & \Lambda_1, \Lambda_2 \vdash s_2 & \end{array}$$

Whether you list the left branch or the right one first when you list the nodes in a derivation makes no difference.

So the rules are to be understood in such a way that if they require the presence of multiple sequents in a series, the order in which the sequents appear is irrelevant.

We saw earlier that sometimes making an assumption is fruitful. In our proof system, you may make any assumption you please at any point—whether the assumption you make is indeed fruitful is another matter, of course. We will treat this as an inference rule. You may:

3.8.2 Assumption Introduction

Assumption Introduction (A) Infer $s \vdash s$.

Of course, you are not really inferring anything when you use this rule since it makes no difference what went before. But we'll call it an inference rule for convenience and uniformity. Consider the lines 4 through 6 of the argument in Section 3.5. We can now add the inference rule to the annotations:

4.	S	$\vdash S$	.....A
5.	Γ	$\vdash S \supset \neg E$	..... premise
6.	Γ, S	$\vdash \neg E$	.....4,5,⊃E

Line 6 is arrived at via ⊃E and that rule requires us to write Γ, S in the datum of the sequent on line 6 given the way lines 4 and 5 look.

Notice that if you ignore the datum and the turnstile, the derivation looks pretty much like the standardized form we saw earlier. What we are adding is a device for keeping track of how, if at all, claims are supported and how that affects the support for any further conclusions.

Also, we still distinguish between assumptions and premises. An assumption is inferred in accordance with the Assumption Introduction rule, but a premise is inserted into a derivation without being inferred.

We now have stated two rules. We will see some more in the following sections.

3.9 Three Rules for Conjunctions and Disjunctions

3.9.1 Disjunction Elimination

Here is an inference rule that corresponds to the Argument By Cases we discussed earlier. In our formal system it is called Disjunction Elimination:

Disjunction Elimination (∨E) From $\Lambda_1 \vdash s_1 \vee s_2$ and $\Lambda_2, s_1 \vdash s_3$ and $\Lambda_3, s_2 \vdash s_3$, infer $\Lambda_1, \Lambda_2, \Lambda_3 \vdash s_3$.

We can use this to formalize Socrates's argument that death is a good thing that we saw earlier. Let's use the following keys:

S: Death is like an eternal deep sleep.
M: Death is a migration of the soul to another place.
G: Death is a good thing.

Here is Socrates's argument as a derivation:

1.	Γ	$\vdash S \vee M$	premise
2.	Δ	$\vdash S \supset G$	premise
3.	Θ	$\vdash M \supset G$	premise
4.	S	$\vdash S$	A
5.	Δ, S	$\vdash G$	2,4,⊃E
6.	M	$\vdash M$	A
7.	Θ, M	$\vdash G$	3,6,⊃E
8.	Γ, Δ, Θ	$\vdash G$	1,5,7,∨E

(Θ is Greek upper case theta.) This derivation looks a little longer than the English version. The English version seems to go from 1, 2, 3 straight to 8. This formalized version is more explicit. We assume each of the disjuncts in turn and show either assumption supports G and then conclude line 8 using ∨E.

Notice that the succedent is the same on lines 5, 7, and 8. The action is on the datum side. Line 5 tells us that S together with Δ supports G. Line 7 tells us that M together with Θ supports G. So they us tell different things. Crucially, they each tell us that accepting G requires one or the other of the assumptions made on lines 4 and 6. But when we get to line 8, it tells us that G is supported by Γ, Δ, and Θ. The assumptions made on lines 4 and 6 are no longer needed even though the assumptions were used to get to lines 5 and 7 which in turn were used (together with line 1) to get to line 8. The assumptions on lines 4 and 6 have been *discharged*.

Here is the same derivation in tree form:

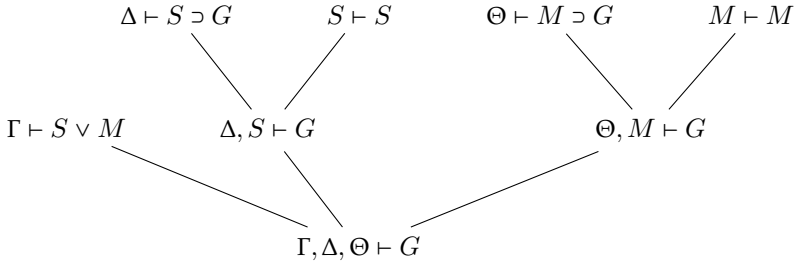


Figure 3.9: Socrates's argument in tree form

The formulation of $\vee$E may suggest that the presence of three sequents is required for the application of $\vee$E but there are special cases where we only need two. We will see that in the exercises.

3.9.2 Disjunction Introduction

There is another rule that has to do with disjunctions. That is the Disjunction Introduction rule:

Disjunction Introduction ($\vee$I) From $\Lambda \vdash s_1$, infer $\Lambda \vdash s_1 \vee s_2$ as well as $\Lambda \vdash s_2 \vee s_1$, for arbitrary s_2.

The truth of $s_1 \vee s_2$ only requires one of the disjuncts to be true. So if Λ supports s_1, it must support $s_1 \vee s_2$ as well as $s_2 \vee s_1$. This rule is called Disjunction *Introduction* because it allows us to infer a sequent whose succedent has

a disjunction as its main connective. Disjunction Introduction may not seem like a useful rule of inference but we will be seeing many fruitful application of this inference rule.

3.9.3 Conjunction Elimination

Here is another inference rule that may strike you as not all that interesting but in fact is of great use:

Conjunction Elimination ($\wedge$E) From $\Lambda \vdash s_1 \wedge s_2$, infer $\Lambda \vdash s_1$ as well as $\Lambda \vdash s_2$.

Surely, if Λ supports $s_1 \wedge s_2$, it follows that that very same evidence supports s_1 as well as s_2. How could anything support the conjunction without supporting each of the conjuncts?

3.10 Sharpening our Understanding of ⊢ and Conjunction Introduction

3.10.1 Conjunction Introduction

I indicated in Section 3.6 that there will be a pair of rules for each connective. Here is the second inference rule concerning conjunction:

Conjunction Introduction (∧I) From $\Lambda_1 \vdash s_1$ and $\Lambda_2 \vdash s_2$, infer $\Lambda_1, \Lambda_2 \vdash s_1 \wedge s_2$.

What it says is that if some consideration supports s_1, and if some possibly different consideration supports s_2, then the two considerations together support $s_1 \wedge s_2$.

Conjunction Introduction may seem unexceptionable. If fossil evidence supports that dinosaurs went extinct 65 million years ago, and if geological evidence supports that the Earth was struck by a meteorite 65 millions ago, then fossil evidence and geological evidence together support that dinosaurs went extinct 65 million years ago and that a meteorite struck the Earth 65 millions ago. Or, more colloquially, the combined evidence supports that a meteorite struck around the same time as when the dinosaurs went extinct 65 million years ago. What could be more boringly obvious than this?

Matters, however, are not so simple. Consider:

1.	Γ	$\vdash s_1$	..... premise
2.	Δ	$\vdash s_2$	..... premise
3.	Γ, Δ	$\vdash s_1 \wedge s_2$	..... 1,2,∧I
4.	Γ, Δ	$\vdash s_1$	..... 3,∧E

The first premise says that some consideration Γ supports s_1. The conclusion says that Γ with some possibly other consideration Δ supports s_1. Notice it does not matter what Δ is. Whatever Δ is, it is bound to support something. So the above argument shows that if Γ supports s_1, no further information can remove the support for s_1. If s_1 is supported *some* things considered, then s_1 is supported *everything* considered. The intuitive notion of support does not work like this. Here is an example to see the point:

> **Example 3.10.1.** During the course of investigating the gunshot murder of an industrialist, inspector Lastrade finds out that the butler had motive—his boss abused him badly over a long period of time—and means—the butler used to be an elite sniper in the army. This supports that the butler killed his boss. Further investigation reveals that the butler was in a different country across the ocean at the time of the murder.

Let Γ capture the evidence that shows the butler had means and motive, and let Δ capture the evidence that shows that the butler was a in a different country at the time of murder. And let B be the claim that the butler killed the industrialist. We have:

$$\Gamma \vdash B.$$

The little derivation from 1 to 4 given above would show that we also have:

$$\Gamma, \Delta \vdash B.$$

But surely that would be a mistake. The combination of Γ and Δ does not support B since that combination shows that the butler lacked opportunity.

The problem is that the fact that the butler has motive and means does not *prove* he did it. That is why some further information can overturn the support for B without calling into question that the butler does have means and motive.

On the other hand, if Lastrade did have genuine proof that the butler did it, then no additional information can change the fact that he has proof. We will, therefore, tighten our interpretation of the turnstile symbol ($\vdash$).

Updated interpretation of $\vdash$: $\Gamma \vdash s$ means that the acceptance of all the claims in Γ *conclusively* supports s. More concisely $\Gamma \vdash s$ means that Γ *proves* s.

Armed with this updated interpretation of the turnstile symbol, we can accept the Conjunction Introduction rule without difficulty.

3.10.2 Monotonicity

Our proof system has the feature that once a set of considerations supports some claim p, then enlarging the set of considerations will not reduce the support for p, because by 'support' we mean 'conclusive support'. A system of reasoning that has this feature is called *monotonic*.

As the Lastrade example (Example 3.10.1) shows, much of our everyday reasoning lacks monotonicity. And this means that our proof system has limitations when it comes to its ability to formalize everyday reasoning. Nevertheless, our proof system captures an important part of our reasoning activities. The most obvious area of reasoning that shows monotonicity is mathematics. And any discipline that employs mathematical tools employs monotonic reasoning at least some of the time.

But we also reason monotonically in everyday life. We often hold fixed certain claims and draw conclusions from those fixed points. That's what all the example arguments except the Lastrade one do. And we often have very good reason to hold those points fixed: they are the things that are well-supported by the evidence and seem safe to be treated as proven. When we formalize arguments that treat certain claims as fixed—they are the premises—we proceed as though there is proof for those claims.

In short, while there are many pieces of reasoning that cannot be captured by our proof system, there are plenty that can be, and some of what can be handled by our system is central to our knowledge generating activities. Do not underestimate the power and scope of what we are doing here.

3.10.3 One more way to rewrite sequents

Since our proof system is monotonic, we will allow adding arbitrary items to the datum of a sequent. E.g., the following is fine:

5.	Γ	$\vdash S$	.....A
6.	Γ, Δ	$\vdash S$	6

In most cases, You could make do without this rule because you could infer from line 5 to what you have on line 6 using A, $\wedge$I, and $\wedge$E. But it is convenient to not have to do that all the time. When you rewrite sequents using this rule, you must be explicit about it.

3.11 Exercises for 3.6 through 3.10

This section is intended to get you used to using sequents in derivations.

1. A derivation is a series of sequents. Because of that, derivations contain information that the standardized presentation of arguments we saw in Section 3.3 do not contain because that style only tracks the succedent side explicitly. The inference rules of our proof system tell you, among other things, how to keep track of things on the datum side.

   Consider the following derivation which is missing the datum on line 3:

   1. Γ $\vdash A \supset B$ ..... premise
   2. Δ $\vdash A$ ..... premise
   3. __ $\vdash B$ ..... 1,2,$\supset$E

   What goes in the datum of line 3? You can see in the annotation that line 3 got there by applying Conditional Elimination to lines 1 and 2. Line 1's datum is Γ and line 2's is Δ. Conditional Elimination tells us that in that case the datum of line 3 is Γ, Δ so that's what goes in there.

   Let's do a couple more of these to get used to paying attention to the datum.

   Fill in the missing datums in the following two derivations:

   (a)
   1. Γ $\vdash (P \vee Q) \supset R$ ..... premise
   2. Δ $\vdash P$ ..... premise
   3. __ $\vdash P \vee Q$ ..... 2,$\vee$I
   4. __ $\vdash R$ ..... 1,3,$\supset$E

   (b)
   1. Γ $\vdash (P \vee Q) \supset R$ ..... premise
   2. __ $\vdash P$ ..... A
   3. __ $\vdash P \vee Q$ ..... 2,$\vee$I
   4. __ $\vdash R$ ..... 1,3,$\supset$E

   (c) You can think of the argument (a) as formalizing something like 'The college catalog says that if Masha has taken logic or has taken

calculus, then she has satisfied the formal reasoning requirement. Her transcript says that she has taken logic. It follows that Masha has satisfied the formal reasoning requirement.' What would argument (b) be formalizing?

2. Our inference rules keep track of *what* supports *what* —the datum is the former what, the succedent the latter. Let's practice using our inference rules to keep track of the succedent side.

   (a) We know that $P \wedge Q$ and $Q \wedge P$ are logically equivalent. That means that if Γ supports $P \wedge Q$ it also supports $Q \wedge P$. Any decent proof system should tells us that, and ours does. Add the missing succedents in the following derivation:

1.	Γ	$\vdash P \wedge Q$	.....premise
2.	Γ	$\vdash$ __	.....1,$\wedge$E
3.	Γ	$\vdash Q$	.....1,$\wedge$E
4.	Γ, Γ	$\vdash$ __	.....2,3,$\wedge$I
5.	Γ	$\vdash Q \wedge P$	4

   Hint: according to the annotations, line 5 is a rewrite of line 4. That tells you what the succedent of line 4 is.

   (b) We also know from Chapter 2 that $P \vee Q$ and $Q \vee P$ are logically equivalent. That means that if Γ supports $P \vee Q$ it also supports $Q \vee P$. Our proof system shows that, too. Add the missing succedents in the following derivation:

1.	Γ	$\vdash P \vee Q$	.....premise
2.	P	$\vdash$ __	.....A
3.	P	$\vdash$ __	.....2,$\vee$I
4.	Q	$\vdash Q$	.....A
5.	Q	$\vdash$ __	.....4,$\vee$I
6.	Γ	$\vdash Q \vee P$	.....1,3,5,$\vee$E

   Notice that there are several lines with exactly the same succedent as the concluding line. An argument in the standardized form would make it much more difficult to discern when one has actually reached the desired conclusion because the standardized form only gives us the succedent side.

3. Comprehending a derivation requires comprehending how the various sequents work together to enable us to infer to the conclusion. Annotations are there to guide our comprehension. In presenting your derivation, it is crucial to make sure that your annotation are correct. Let's practice annotating.

   $P \wedge (Q \vee R)$ and $(P \wedge Q) \vee (P \wedge R)$ are logically equivalent. So if we have evidence for the former sentence, we have evidence for the latter. We can show that. Fill in the missing annotations in the following derivation:

1.	Γ	$\vdash P \wedge (Q \vee R)$	premise
2.	Γ	$\vdash P$	1,__
3.	Γ	$\vdash Q \vee R$	__
4.	Q	$\vdash Q$	__
5.	Γ, Q	$\vdash P \wedge Q$	__
6.	Γ, Q	$\vdash (P \wedge Q) \vee (P \wedge R)$	5,$\vee$I
7.	R	$\vdash R$	__
8.	Γ, R	$\vdash P \wedge R$	__
9.	Γ, R	$\vdash (P \wedge Q) \vee (P \wedge R)$	__
10.	Γ, Γ, Γ	$\vdash (P \wedge Q) \vee (P \wedge R)$	__
11.	Γ	$\vdash (P \wedge Q) \vee (P \wedge R)$	__

4. Fill in the missing datums, succedents, annotations in the following derivation from $\Gamma \vdash (P \vee Q) \vee R)$ to $\Gamma \vdash P \vee (Q \vee R)$.

1.	Γ	$\vdash (P \vee Q) \vee R$	premise
2.	__	$\vdash P \vee Q$	A
3.	P	$\vdash P$	A
4.	__	$\vdash P \vee (Q \vee R)$	3,$\vee$I
5.	Q	$\vdash$ __	A
6.	__	$\vdash Q \vee R$	5,$\vee$I
7.	Q	$\vdash$ __	6,$\vee$I
8.	$P \vee Q$	$\vdash P \vee (Q \vee R)$	2,4,7,$\vee$E
9.	R	$\vdash R$	A

10. R $\vdash Q \vee R$ ..... 9,∨I
11. R $\vdash P \vee (Q \vee R)$ ..... __
12. Γ $\vdash P \vee (Q \vee R)$ ..... 1,8,11,∨E

5. I mentioned that ∨E need not require three sequents. Here is an example of that:

    1. Γ $\vdash P \vee P$ ..... premise
    2. Δ, P $\vdash Q$ ..... premise
    3. Γ, Δ $\vdash Q$ ..... 1,2,2,∨E

    Notice that in the annotation of the third line, line 2 is referred to twice. It is used once as $\Lambda_2, s_1 \vdash s_3$ and again as $\Lambda_2, s_2 \vdash s_3$. This is possible because two of the sequents that must be matched for using ∨E have the same form. You can do the same using ∧I to derive from $\Gamma \vdash P$ to $\Gamma \vdash P \wedge P$. Add the missing annotations:

    1. Γ $\vdash P$ ..... __
    2. Γ $\vdash P \wedge P$ ..... __

6. Construct the following derivations:

    (a) From $\Gamma \vdash Q \wedge P$ to $\Gamma \vdash Q \vee P$.

    (b) From $\Gamma \vdash \neg Q \wedge \big(Q \wedge P\big)$ to $\Gamma \vdash Q$.

    (c) From $\Gamma \vdash P$ to $\Gamma \vdash \big(P \wedge P\big) \vee Q$

3.12 Rules for Negation

3.12.1 Negation Elimination

Here is an obvious inference rule:

Negation Elimination (¬E) From $\Lambda \vdash \neg\neg s$, infer $\Lambda \vdash s$.

Double negation is affirmation. Strictly speaking, the rule should be called *Double* Negation Elimination but we'll omit the 'double' to save a word.

3.12.2 Negation Introduction

With double negation out of the way, let's look at something more interesting. That's the rule corresponding to Reductio ad Absurdum we saw earlier. The version in our proof system is called Negation Introduction:

Negation Introduction (¬I) From $\Lambda_1, s_1 \vdash s_2$ and $\Lambda_2, s_1 \vdash \neg s_2$, infer $\Lambda_1, \Lambda_2 \vdash \neg s_1$.

We can use this to represent as a derivation the Argument 3 in Section 3.3 that you cannot go back in time and kill your mother before you were even conceived. Let S mean that your plan succeeds, and E that you exist:

1.	Γ	$\vdash S \supset E$	premise
2.	Δ	$\vdash S \supset \neg E$	premise
3.	S	$\vdash S$	A
4.	Γ, S	$\vdash E$	1,3,⊃E
5.	Δ, S	$\vdash \neg E$	2,3,⊃E
6.	Γ, Δ	$\vdash \neg S$	4,5,¬I

As with the case of ∨I, this derivation looks longer than the English version which seems to go from lines 1 and 2 straight to line 6. This formal version

is more explicit. Notice that you could use ∧I to get from line 4 and 5 to a sequent with an outright contradiction as succedent: $\Gamma, \Delta, S \vdash P \wedge \neg P$. Our derivation is much more explicit about this threat of contradiction than the English version of Reductio Ad Absurdum we saw earlier.

Here is an interesting point about Negation Introduction: we can use it to achieve the same result as Modus Tollens. Modus Tollens, if you recall, starts with a conditional and the negation of its consequent as premises and concludes with the negation of the antecedent of the conditional. While some systems have an inference rule corresponding to Modus Tollens as a separate inference rule, our system does not. We can infer from $\Gamma \vdash P \supset Q$ and $\Delta \vdash \neg Q$ to $\Gamma, \Delta \vdash \neg P$ with the rules we already have:

7.	Γ	$\vdash P \supset Q$	premise
8.	Δ	$\vdash \neg Q$	premise
9.	P	$\vdash P$	A
10.	Γ, P	$\vdash Q$	7,9,⊃E
11.	Δ, P	$\vdash \neg Q$	8
12.	Γ, Δ	$\vdash \neg P$	10,11,¬I

As you can see, we just need A, ⊃E, and ¬I to achieve the same effect as Modus Tollens.

3.13 Conditional Introduction

When we discussed common inference patterns, we saw a way of concluding with a conditional. The last rule of our proof system corresponds to that move and is called Conditional Introduction:

Conditional Introduction (⊃I) From $\Lambda, s_1 \vdash s_2$, infer $\Lambda \vdash s_1 \supset s_2$.

This is surely right. If Λ and s_1 together conclusively support s_2, then Λ must conclusively support that $s_1 \supset s_2$.

We can now formalize the argument at the end of 3.4 as a derivation—that was the argument that prompted us to switch to using sequents:

1.	S	$\vdash S$	A
2.	Γ	$\vdash S \supset \neg E$	premise
3.	Γ, S	$\vdash \neg E$	1,2,⊃E
4.	Δ	$\vdash \neg E \supset \neg P$	premise
5.	Γ, Δ, S	$\vdash \neg P$	3,4,⊃E
6.	Γ, Δ	$\vdash S \supset \neg P$	5,⊃I

This concludes the presentation of the inference rules.

3.14 Proof System

We have seen several rules of inference. They specify how you may extend a series of sequents. You do not have to extend a series, but if you do, you must do it in accordance with these only: everything is forbidden—*alles verboten* as one says in German—except for moves that are explicitly permitted by these rules. Here they are in one place.

You may:

Assumption Introduction (A) Infer $s \vdash s$.

Conjunction Introduction ($\wedge$I) From $\Lambda_1 \vdash s_1$ and $\Lambda_2 \vdash s_2$, infer $\Lambda_1, \Lambda_2 \vdash s_1 \wedge s_2$.

Conjunction Elimination ($\wedge$E) From $\Lambda \vdash s_1 \wedge s_2$, infer $\Lambda \vdash s_1$ as well as $\Lambda \vdash s_2$.

Disjunction Introduction ($\vee$I) From $\Lambda \vdash s_1$, infer $\Lambda \vdash s_1 \vee s_2$ as well as $\Lambda \vdash s_2 \vee s_1$, for any s_2.

Disjunction Elimination ($\vee$E) From $\Lambda_1 \vdash s_1 \vee s_2$ and $s_1, \Lambda_2 \vdash s_3$ and $s_2, \Lambda_3 \vdash s_3$, infer $\Lambda_1, \Lambda_2, \Lambda_3 \vdash s_3$.

Negation Introduction ($\neg$I) From $\Lambda_1, s_1 \vdash s_2$ and $\Lambda_2, s_1 \vdash \neg s_2$, infer $\Lambda_1, \Lambda_2 \vdash \neg s_1$.

Negation Elimination ($\neg$E) From $\Lambda \vdash \neg\neg s$, infer $\Lambda \vdash s$.

Conditional Introduction ($\supset$I) From $\Lambda, s_1 \vdash s_2$, infer $\Lambda \vdash s_1 \supset s_2$.

Conditional Elimination ($\supset$E) From $\Lambda_1 \vdash s_1 \supset s_2$ and $\Lambda_2 \vdash s_1$, infer $\Lambda_1, \Lambda_2 \vdash s_2$.

Also, you may rewrite the datum side of sequents in the following ways:

a. You may reorder items within the datum as you see fit.

b. You may delete duplicate items within the datum.

c. You may add arbitrary items to the datum of a sequent.

These rules together form a proof system. You can use them to construct all sorts of proofs.

3.15 Exercises for 3.12 through 3.14

1. Here is something obvious. If we have evidence that $P \vee Q$ and we have evidence that $\neg P$, we have evidence that Q. Our proof system confirms this. Add the missing datums in the following derivation:

1.	Γ	$\vdash P \vee Q$	premise
2.	Δ	$\vdash \neg P$	premise
3.	P	$\vdash P$	A
4.	__	$\vdash \neg P$	2
5.	__	$\vdash P$	3
6.	__	$\vdash \neg\neg Q$	4,5,¬I
7.	Δ, P	$\vdash Q$	6,¬E
8.	Q	$\vdash Q$	A
9.	__	$\vdash Q$	1,7,8,∨E

2. Derivations can often be adapted to prove something similar. For instance, the above can be adapted easily to derive from $\Gamma \vdash \neg P \vee Q$ and $\Delta \vdash P$ to $\Gamma, \Delta \vdash Q$. Fill in the missing datums and annotations.

1.	Γ	$\vdash \neg P \vee Q$	premise
2.	Δ	$\vdash P$	premise
3.	__	$\vdash \neg P$	A
4.	__	$\vdash P$	2
5.	__	$\vdash \neg P$	3
6.	__	$\vdash \neg\neg Q$	4,5,¬I
7.	__	$\vdash Q$	6,¬E
8.	Q	$\vdash Q$	A
9.	__	$\vdash Q$	1,7,8,∨E

3. The Greek capital letters on the datum side are place-holders. You can plug anything you want into them. Take the derivation in Problem 2 above. You can plug P into Δ and add one more step at the end to show that you can infer from $\Gamma \vdash \neg P \vee Q$ to $\Gamma \vdash P \supset Q$ (which we should expect given the way the conditional is defined). Construct such a derivation.

4. The following is a derivation from $\Gamma \vdash P \supset Q$ to $\Gamma \vdash \neg P \vee Q$. Again, we should expect that there is such a derivation given the way the conditional was defined. Fill in the missing parts of the derivation:

1.	Γ	$\vdash P \supset Q$	..... premise
2.	$\neg(\neg P \vee Q)$	$\vdash$ __	..... A
3.	__	$\vdash \neg P$	..... A
4.	$\neg P$	$\vdash \neg P \vee Q$	..... __
5.	__	$\vdash$ __	 2
6.	$\neg(\neg P \vee Q)$	$\vdash \neg\neg P$	..... 4,5,¬I
7.	__	$\vdash$ __	..... 6,¬E
8.	__	$\vdash$ __	..... 1,7,⊃E
9.	$\Gamma, \neg(\neg P \vee Q)$	$\vdash \neg P \vee Q$	..... 8,∨I
10.	Γ	$\vdash \neg\neg(\neg P \vee Q)$	..... __
11.	Γ	$\vdash \neg P \vee Q$	..... __

5. Suppose there is evidence that $P \supset Q$. In that case, there is is evidence that $\neg Q \supset \neg P$. Our proof system confirms this. Construct a derivation from $\Gamma \vdash P \supset Q$ to $\Gamma \vdash \neg Q \supset \neg P$. Hint: you can adapt and modify one the derivations given in Section 3.12.

6. If you have evidence that $(X \vee Y) \supset Z$, you have evidence that $X \supset Z$. Construct a derivation from $\Gamma \vdash (X \vee Y) \supset Z$ to $\Gamma \vdash X \supset Z$.

Chapter 4

Proofs and Truth

4.1 Sentential Logic and Its Theorems

So far, the derivations we have seen had premises. Let us move to derivations that have no premises. You might wonder how there could be derivations without premises. Don't we need some starting points and aren't they the premises? But in our proof system it's easy to construct a derivation without premises. Consider:

1. p	$\vdash p$	..... A
2.	$\vdash p \supset p$	..... 1,⊃I

There are two points to note about this derivation. First, it has no premises—just check the annotations. Secondly, the concluding sequent has an empty datum. Let me discuss them in order.

The derivation has no premises. But it does have an assumption. A premise-less derivation is a derivation that has no premises. But that does not mean it cannot have assumptions. The above is an example of that. And notice that we do have an inference rule for getting to our assumptions: the Assumption Introduction rule. So a premise-less derivation can be constructed using nothing more than our rules of inference. The typical premises of the derivations in the previous sections require some work to get hold of—often that work is empirical research as well as some philosophical reflection. But premise-less derivations require no appeals whatever to knowledge generated outside of our proof system. Such derivations are of particular interest to the study of logic because it means that there are things we can figure out via logic independently of any other sources of knowledge. We will call our proof system combined with our characterization of a formal language $\mathcal{L}$ a *system of sentential logic*, or just *sentential logic* for short. In this chapter, we will be exploring important features of sentential logic.

We will call a premise-less derivation a *proof*, and when we can derive a sequent through a proof, we say that we have proved the sequent. So a multi-line proof proves each sequent that appears in the proof. Since a derivation can only be of finite length, the same applies to proof.

Now to the second point of interest. The concluding sequent, line 4, has an empty datum. When we can prove a sequent that has an empty datum, we call the succedent of the sequent a *theorem* of sentential logic. And in such cases, we will also say that we have proved the theorem. Notice that what's called

the theorem is the succedent of the relevant sequent. That is, a theorem is a sentence of $\mathcal{L}$ and not a sequent. What does such a theorem tell us?

A sequent $\Gamma \vdash s$ means that Γ conclusively supports s. So if we accept all the claims in Γ, we must accept the claim that s. In the case of a theorem t, we can prove $\Gamma \vdash t$ there Γ is empty. This means means that we must accept s regardless of whatever else we accept. That is, we must accept s regardless of what we believe and know about the world.

Some theorems are more interesting than others. The more interesting ones often have commonly used names. The above theorem tells us that it's a theorem that a sentence implies itself. It is known as **Identity (ID)**.

Conditional Introduction is an easy way of inferring to a theorem. The reason it works is that when you apply the inference rule, the number of items in the datum decreases by one. Conditional Introduction is not the only inference rule with this feature. Negation Introduction also has this feature. Recall:

Negation Introduction (¬I) From $\Gamma, s_1 \vdash s_2$ and $\Delta, s_1 \vdash \neg s_2$, infer $\Gamma, \Delta \vdash \neg s_1$.

We start with three items, Γ, Δ, and s_1 on the datum side and end with only two: Γ, Δ. This means that $\neg I$ can also be used to prove a theorem. Here is an example:

1.	$p \wedge \neg p \vdash p \wedge \neg p$	A
2.	$p \wedge \neg p \vdash p$	1,∧E
3.	$p \wedge \neg p \vdash \neg p$	1,∧E
4.	$\vdash \neg(p \wedge \neg p)$	2,3,¬I

Given the contradiction $p \wedge \neg p$, its negation is a theorem of sentential logic. This is known as **Non-Contradiction (NC)**.

Here is another important theorem closely related to Non-Contradiction. Non-Contradiction says that we can prove that p is not both true and not-true. We can also prove that p is either true or not-true—that is, it cannot be that p is neither true nor false. This is known as **Excluded Middle (EM)**. Here is a proof:

1.	$\neg(p \vee \neg p)$	$\vdash \neg(p \vee \neg p)$	A
2.	p	$\vdash p$	A
3.	p	$\vdash p \vee \neg p$	2,$\vee$I
4.	$p, \neg(p \vee \neg p)$	$\vdash p \vee \neg p$	3
5.	$\neg(p \vee \neg p), p$	$\vdash \neg(p \vee \neg p)$	1
6.	$\neg(p \vee \neg p)$	$\vdash \neg p$	4,5,$\neg$I
7.	$\neg(p \vee \neg p)$	$\vdash p \vee \neg p$	6,$\vee$I
8.		$\vdash \neg\neg(p \vee \neg p)$	1,7,$\neg$I
9.		$\vdash p \vee \neg p$	8,$\neg$E

Notice that p in the proofs of the theorems above is a sentential variable. So strictly speaking the proofs are templates of proofs. And a theorem is not a particular sentence of $\mathcal{L}$ but a sentence template. You can plug any actual sentence of $\mathcal{L}$ into p and get an actual proof of a sentence. A theorem of sentential logic, therefore, tells us that any sentence of a particular form can be derived using the derivation rules of our proof system without needing any extra knowledge. From now on, we will mainly be discussing such proof templates—but we will call them proofs for brevity.

In order to avoid confusions, we will make the following stipulation: you are allowed to derive a sequent with an empty datum only if you can do so without depending on a premise. This ensures that if we can derive $\vdash s$, s is a theorem.

4.2 Working with Proof Templates

Given a proof template, you can replace a variable with any other variable and that will make no difference so long as you make sure that you replace consistently throughout. For instance, take the first proof from the previous section:

1. p $\vdash p$ . . . . . A
2. $\vdash p \supset p$ . . . . . 1,⊃I

You can replace p with s throughout to get:

3. s $\vdash s$ . . . . . A
4. $\vdash s \supset s$ . . . . . 3,⊃I

You can also plug complex sentences into p. E.g., you could plug $\neg A$ into p and get:

5. $\neg A$ $\vdash \neg A$ . . . . . A
6. $\vdash \neg A \supset \neg A$ . . . . . 5,⊃I

You can plug the negation of any sentence into p and get a fine proof. That means you can replace p with $\neg q$ and get another fine proof template:

7. $\neg q$ $\vdash \neg q$ . . . . . A
8. $\vdash \neg q \supset \neg q$ . . . . . 7,⊃I

In fact, you can replace p with any complex sentence and that means that you can replace p with any complex sentence template. E.g., the following is fine:

9. $q \wedge s$ $\vdash q \wedge s$ . . . . . A
10. $\vdash (q \wedge s) \supset (q \wedge s)$ . . . . . 9,⊃I

Such systematic replacements of variables make no difference to the proof—just think about how the inference rules work. So you may rewrite theorems by replacing sentence variables with other sentence variables or with sentence templates. This will make life easier in what follows.

Be careful that you only make replacements and not engage in illicit derivations. For instance, you can do:

11.	$\neg\neg r$	$\vdash \neg\neg r$	A
12.		$\vdash \neg\neg r \supset \neg\neg r$	11,⊃I

But you may ***not*** delete the double negations in line 12 unless you can prove that that is ok (it's not that difficult to prove, but you absolutely must prove it).

4.3 More Theorems of Sentential Logic

Here is a theorem known as **DeMorgan's Law (DM)** (there are a few other theorems known by the same name):

1.	$\neg(p \vee q)$	$\vdash \neg(p \vee q)$	A
2.	$\neg(p \vee q), p$	$\vdash \neg(p \vee q)$	1
3.	p	$\vdash p$	A
4.	p	$\vdash p \vee q$	3,$\vee$I
5.	$\neg(p \vee q)$	$\vdash \neg p$	2,4,$\neg$I
6.	$\neg(p \vee q), q$	$\vdash \neg(p \vee q)$	1
7.	q	$\vdash q$	A
8.	q	$\vdash p \vee q$	7,$\vee$I
9.	$\neg(p \vee q)$	$\vdash \neg q$	6,8,$\neg$I
10.	$\neg(p \vee q)$	$\vdash \neg p \wedge \neg q$	5,9,$\wedge$I
11.		$\vdash \neg(p \vee q) \supset (\neg p \wedge \neg q)$	10,$\supset$I

Theorems can be proven from assumptions only and that means we could insert the proof of a theorem into any other proof without disturbing it. And that can be useful. In order to save space, time, and energy, you do not have to restate the proof of a theorem provided you have already proven it somewhere. We name theorems that we have proven, and in the annotation we just state the name of the theorem. The proof of the following theorem, **Implication (IM)** is an example:

12.	$p \supset q$	$\vdash p \supset q$	A
13.	$\neg(\neg p \vee q)$	$\vdash \neg(\neg p \vee q)$	A
14.		$\vdash \neg(\neg p \vee q) \supset (\neg\neg p \wedge \neg q)$	DM
15.	$\neg(\neg p \vee q)$	$\vdash \neg\neg p \wedge \neg q$	13,14,$\supset$E

16. $\neg(\neg p \vee q) \vdash \neg\neg p$ . . . . . 15,$\wedge$E
17. $\neg(\neg p \vee q) \vdash p$ . . . . . 16,$\neg$E
18. $p \supset q, \neg(\neg p \vee q) \vdash q$ . . . . . 12,17,$\supset$E
19. $\neg(\neg p \vee q) \vdash \neg q$ . . . . . 15,$\wedge$E
20. $p \supset q \vdash \neg\neg(\neg p \vee q)$ . . . . . 18,19,$\neg$I
21. $p \supset q \vdash \neg p \vee q$ . . . . . 20,$\neg$E
22. $\vdash (p \supset q) \supset (\neg p \vee q)$ . . . . . 21,$\supset$I

On line 14 we just state the theorem without giving its proof—we could, but it would be a lot of work. The annotation tells us where to look for the proof. Note that we replaced p in our proof of DeMorgan with $\neg p$.

Given Implication, we can give a shorter looking proof of Excluded Middle which may be easier to see through given the definition of $\supset$:

23. $\vdash p \supset p$ . . . . . ID
24. $\vdash (p \supset p) \supset (\neg p \vee p)$ . . . . . IM
25. $\vdash \neg p \vee p$ . . . . . 23,24,$\supset$E

This proof looks short because instead of giving an explicit proof of Implication, we merely state the theorem. Appealing to theorems proved elsewhere is a real time and space saver.

4.3.1 Some useful theorems

Here are some theorems that are often useful. Most of the names are commonly used ones. Notice that several related but distinct theorems have the same name (e.g., DeMorgan).

If you find yourself appealing to these theorems, use the name of the theorem in your annotations:

Identity (ID)
$p \supset p$

Non-Contradiction (NC)
$\neg(p \wedge \neg p)$

Excluded Middle (EM)
$p \vee \neg p$

DeMorgan (DM)
$\neg(p \vee q) \supset (\neg p \wedge \neg q)$

DeMorgan (DM)
$(\neg p \wedge \neg q) \supset \neg(p \vee q)$

DeMorgan (DM)
$\neg(p \wedge q) \supset (\neg p \vee \neg q)$

DeMorgan (DM)
$(\neg p \vee \neg q) \supset \neg(p \wedge q)$

Implication (IM)
$(p \supset q) \supset (\neg p \vee q)$

Elimination (EL)
$(p \vee q) \supset (\neg p \supset q)$

Contraposition (CP)
$(p \supset q) \supset (\neg q \supset \neg p)$

Commutativity of Conjunction (CC)
$(p \wedge q) \supset (q \wedge p)$

Commutativity of Disjunction (CD)
$(p \vee q) \supset (q \vee p)$

Associativity of Conjunction (AC)
$[(p \wedge q) \wedge r] \supset [p \wedge (q \wedge r)]$

Associativity of Conjunction (AC)
$[p \wedge (q \wedge r)] \supset [(p \wedge q) \wedge r]$

Associativity of Disjunction (AD)
$[(p \vee q) \vee r] \supset [p \vee (q \vee r)]$

Associativity of Disjunction (AD)
$\big[p \vee \big(q \vee r\big)\big] \supset \big[\big(p \vee q\big) \vee r\big]$

Double Negation Introduction (DN)
$p \supset \neg\neg p$

4.4 Using Theorems in Derivations

Theorems can be used to make proofs of other theorems easier. But there is nothing special about proofs of theorems in this regard. We can also use theorems in more general derivations that we discussed in the previous chapter. Appealing to theorems can make certain derivations much shorter and intuitive. Consider, for instance, the following formalization of an argument that in the plain English version uses Modus Tollens:

1.	Γ	$\vdash P \supset Q$	premise
2.	Δ	$\vdash \neg Q$	premise
3.		$\vdash (P \supset Q) \supset (\neg Q \supset \neg P)$	CP
4.	Γ	$\vdash \neg Q \supset \neg P$	1,3,$\supset$E
5.	Γ, Δ	$\vdash \neg P$	2,4,$\supset$E

This is a little shorter than the derivation we gave in Section 3.12 and it might feel like a more natural representation of how we reason using Modus Tollens.

Our proof system is designed to use only a very small number of inference rules and that often results in surprisingly long formal derivations of matters that seem obvious. Appealing to theorems can make things a lot shorter and easier to see through by hiding some of the complexity. Here is another example, deriving from $\Gamma \vdash P \vee Q$ and $\Delta \vdash \neg P$ to $\Gamma, \Delta \vdash \neg Q$:

1.	Γ	$\vdash P \vee Q$	premise
2.	Δ	$\vdash \neg P$	premise
3.		$\vdash (P \vee Q) \supset (\neg P \supset Q)$	EL
4.	Γ	$\vdash \neg P \supset Q$	1,3,$\supset$E
5.	Γ, Δ	$\vdash Q$	2,4,$\supset$E

This should be much easier to comprehend than the derivation we saw earlier in one of our exercises.

4.5 Valid Arguments

A very important idea in discussions of arguments is that of *valid* arguments that we mentioned earlier. An argument is said to be valid iff. it is impossible for the conclusion to be false while all the premises are true. In other words, in a valid argument, the truth of all the premises guarantees the truth of the conclusion. Valid arguments are of great interest because given reasons to accept the premises of a valid argument, we will thereby have reason to accept its conclusion: you cannot rationally accept the premises while rejecting the conclusion of a valid argument—they leave no such wiggle room.

Consider a very simple derivation:

1. Γ	$\vdash P \supset Q$	..... premise
2. Δ	$\vdash P$	..... premise
3. Γ, Δ	$\vdash Q$	..... 1,2,$\supset$E

Given our interpretation of a sequent, what this derivation tells us is that given conclusive support for $P \supset Q$ and P, there is also conclusive support for Q. That is, the derivation tells us what would be true if an argument from $P \supset Q$ and P to Q were valid. But would such an argument be valid? Of course, it would be. We can see that by looking at the truth table of the conditional. Just look at the first row (both the conditional and its antecedent are true) of:

s_1	s_2	$(s_1 \supset s_2)$
T	T	T
F	T	T
T	F	F
F	F	T

Here is a question: given a successful derivation from $\Gamma_1 \vdash P_1$ through $\Gamma_n \vdash P_n$ to $\Gamma_1, \ldots, \Gamma_n \vdash Q$, is there a valid argument from P_1, ..., P_n to Q? That is, does the truth of all of P_1 through P_n guarantee the truth of Q given the existence of the derivation?

Here is a related question. Suppose the truth all of P_1 through P_n guarantees the truth of Q. Is there a derivation from $\Gamma_1 \vdash P_1$ through $\Gamma_n \vdash P_n$ to $\Gamma_1, \ldots, \Gamma_n \vdash Q$?

We want the answers to both of these questions to be Yes. Because in that case, we can say that: a) any argument that can be formalized as a derivation in our proof system is valid; and b) any valid argument can be captured by a derivation in our proof system.

We will see in the next few sections that the answer to both these questions is indeed Yes.

Some preliminary exercises:

1. Suppose there is a derivation from $\Gamma \vdash s$ to $\Gamma \vdash t$. Explain why it follows that $\vdash s \supset t$. (Hint: you can plug anything you want into Γ.)

2. Suppose $s \supset t$ is a tautology. Explain why in that case the truth of s guarantees the truth of t.

3. Suppose we can show that if $\vdash s \supset t$, then $s \supset t$ is a tautology. Explain why this would show that if there is a derivation from $\Gamma \vdash s$ to $\Gamma \vdash t$, then the truth of s guarantees the truth of t.

4. Suppose we can show that if $s \supset t$ is a tautology, then $\vdash s \supset t$. Explain why this would show that if $s \supset t$ is a tautology, then there is a derivation from $\Gamma \vdash s$ to $\Gamma \vdash t$.

4.6 Theorems and Tautologies

Let us introduce some terminology and symbols to make our life easier in answering the questions raised in the previous section.

When the truth of one sentence p guarantees the truth of another sentence q, we say that p *entails* q. We write

$$p \vDash q$$

to express that p entails q. The entailment symbol, $\vDash$ (double turnstile), is *not* part of $\mathcal{L}$. It is a symbol we use in our metalanguage to concisely express that the truth of one sentence guarantees the truth of another. Truth tables tell us that the truths of several sentences together guarantee the truth of another. For instance, the truth table for disjunction tells that if $p \vee q$ and $\neg p$ are both true, that guarantees that q is true. In such a case, we use a comma separated list on the left side of the entailment sign to indicate the truths of all those sentences on the list guarantees the truth of the sentence on the right of the entailment sign:

$$p \vee q, \neg p \vDash q$$

The order in which we list the sentences on the left is obviously irrelevant. Notice that if $p \vDash q$, then p's truth alone suffices to guarantee the truth of q which means that it does not matter what else is, or isn't, true. Thus, if $p \vDash q$, then $p, \Gamma \vDash q$ must also hold for any list of sentences Γ.

Suppose t is a tautology. What is the list of sentences whose truths guarantees the truth of t? Since t is a tautology, any list of sentences is such that the truth of all the sentences on the list guarantees the truth of t. To indicate that it does not matter what list goes on the left side of the entailment symbol, we leave things blank there and write:

$$\vDash t$$

to mean that t is a tautology.

You will have noticed some obvious similarities between the uses of $\vdash$ and $\vDash$ symbols. But the two symbols mean different things. $\vdash$ concerns what we can

reason our way to, and $\models$ is about what is true. For instance, inspecting the truth table of a sentence t might tell you that it is a tautology, but so far you have no reason to think that you will be able to construct a derivation of $\vdash t$ (take a look again at the tautologies in Section 2.6 and see if you can prove them in our proof system).

With this notation in place, let's consider generalized versions of the preliminary exercises of the last section:

1. Explain why there is a derivation from $\Gamma_1 \vdash s_1$, $\Gamma_2 \vdash s_2$, ..., $\Gamma_n \vdash s_n$ to $\Gamma_1, \Gamma_2, \ldots, \Gamma_n \vdash t$ iff. $\vdash [s_1 \wedge (s_2 \ldots \wedge s_n)] \supset t$. (imagine all the brackets in place to make things well-formed.)

2. Explain why $\models (s_1 \wedge s_2 \ldots \wedge s_n) \supset t$ iff. $s_1, s_2, \ldots, s_n \models t$.

3. Suppose we can show that if $\vdash [s_1 \wedge (s_2 \ldots \wedge s_n)] \supset t$, then $\models (s_1 \wedge s_2 \ldots \wedge s_n) \supset t$. Explain why this would show that: if there is a derivation from $\Gamma_1 \vdash s_1$, $\Gamma_2 \vdash s_2$, ..., $\Gamma_n \vdash s_n$ to $\Gamma \vdash t$, then $s_1, s_2, \ldots, s_n \models t$.

4. Suppose we can show that if $\models (s_1 \wedge s_2 \ldots \wedge s_n) \supset t$, then $\vdash [s_1 \wedge (s_2 \ldots \wedge s_n)] \supset t$. Explain why this would show that: if $\models (s_1 \wedge s_2 \ldots \wedge s_n) \supset t$, then there is a derivation from $\Gamma_1 \vdash s_1$, $\Gamma_2 \vdash s_2$, ..., $\Gamma_n \vdash s_n$ to $\Gamma_1, \Gamma_2, \ldots, \Gamma_n \vdash t$.

If we can show our suppositions in 3. and 4. above, we can show that an argument can be captured by a derivation if and only if it is valid. And we can show those two things, if we can show the following two more general claims:

Soundness. If $\vdash t$, then $\models t$.

Completeness. If $\models t$, then $\vdash t$.

We will prove both of these claims in what follows.

4.7 Soundness

Let's start with soundness. When a proof system is such that all its theorems are tautologies, the proof system is called *sound*. We want to show that our proof system is sound.

To make stating our argument easier, let's say say that a sequent $\Gamma \vdash s$ is *correct* iff. $\Gamma \vDash s$ (in particular, $\vdash s$ is correct iff. $\vDash s$).

Now the argument showing the soundness of our proof system. Recall that a proof is constructed by extending a series of sequents in accordance with the inference rules. We will show the following two points about such a series of sequents:

(i) If the series has only one sequent, that sequent is correct.

(ii) If all the sequents of a series of *n* sequents are correct, then extending the series by one sequent in accordance with the inference rules results in a series of *n+1* correct sequents.

If both of these points hold, then any series of length 1 consists of a correct sequent (because of (i)). So, by (ii), any series of length 2 consists of only correct sequents; ditto for any series of lengths 3, 4, etc. for any natural number n. So any series of sequents constructed in accordance with the inference rules consists of only correct sequents. In particular, if any sequent appearing in such a series has an empty datum, its succedent is a tautology. So any theorem is a tautology.

Let's deal with the two points one by one.

4.7.1 Series has only one sequent

(i) is trivial. If a series of sequents constructed in accordance with the inference rules has length 1, that single sequent must have been inferred using the Assumption Introduction rule because that is the only rule that does not require any preceding sequents. And the one sequent must take the form $s \vdash s$. Moreover, $s \vDash s$. So (i) is true.

4.7.2 Series has only correct sequents up to n-th line

What about (ii)? This is more work, but we have in fact done most of the grunt work already. Most of the task is putting together the various pieces.

We will say that an inference rule is *valid* just in case it is such that: if all previous sequents are correct, then applying the rule results in another correct sequent.

What we want to show is that all of our rules of inference are valid. If they are, then if all the sequents in a series of *n* sequents are correct, so is a sequent inferred from that series.

We will work through the rules one by one.

1. **Assumption Introduction**

   The Assumption Introduction rule is obviously valid since it infers to $s \vdash s$ and that is correct. In particular, it is correct if all the previous sequents are correct.

2. **Conditional Introduction**

   Applying Conditional Introduction requires the presence of a sequent of the form

   $\Gamma, s_1 \vdash s_2$

   and allows inferring from this the sequent

   $\Gamma \vdash s_1 \supset s_2$.

   We are assuming that all sequents prior to applying this rule are correct. That is, we assume that $\Gamma, s_1 \vDash s_2$. We need to show that given this assumption $\Gamma \vdash s_1 \supset s_2$ is also correct; i.e., $\Gamma \vDash s_1 \supset s_2$.

   Now, suppose the truth of Γ and the truth of s_1 together guarantee the truth of s_2 ('truth of Γ' is short for 'truth of all the sentences of Γ). Further suppose that Γ on its own does not guarantee the truth of $s_1 \supset s_2$. That means that Γ could be true, and s_1 true, and s_2 false. But this contradicts the first supposition that the truth of Γ and the truth s_1 together guarantee the truth of s_2. Thus, the truth of Γ guarantees the truth of $s_1 \supset s_2$:

   If $\Gamma, s_1 \vDash s_2$, then $\Gamma \vDash s_1 \supset s_2$.

Comparing with the Conditional Introduction rule, we can see that:

$$\text{if } \Gamma, s_1 \vdash s_2 \text{ is correct then so is } \Gamma \vdash s_1 \supset s_2.$$

Thus, Conditional Introduction is valid.

3. **Conditional Elimination**

   According to the truth table of the conditional, given a true conditional, if the antecedent of a conditional is true, its consequent is also true. So if the antecedent of a true conditional is guaranteed to be true, then its consequent is guaranteed to be true. Thus, if Γ guarantees the truth of $s_1 \supset s_2$ and Δ guarantees the truth of s_1, then Γ, Δ must guarantee the truth of s_2:

   $$\text{If } \Gamma \vDash s_1 \supset s_2 \text{ and } \Delta \vDash s_1, \text{ then } \Gamma, \Delta \vDash s_2.$$

   So Conditional Elimination is a valid rule of inference.

   There is a special case of the point we just made that we will be exploiting:

   $$\text{If } \vDash s_1 \supset s_2 \text{ and } \Gamma \vDash s_1, \text{ then } \Gamma \vDash s_2.$$

   Or, more usefully:

   $$\text{If } \vDash s_1 \supset s_2, \text{ then } \langle \text{if } \Gamma \vDash s_1 \text{ then } \Gamma \vDash s_2 \rangle.$$

   What this means is: if $\vDash s_1 \supset s_2$, then any rule that tells us that we may infer from $\Gamma \vdash s_1$ to $\Gamma \vdash s_2$ is valid.

4. **Negation Elimination**

   We have already shown that $\vDash \neg\neg s \supset s$. Thus, by what we have just pointed out above, Negation Elimination is valid.

5. **Disjunction Introduction**

   We also have shown that $\vDash s_1 \supset (s_1 \vee s_2)$ as well as $\vDash s_1 \supset (s_2 \vee s_1)$. Thus Disjunction Introduction is valid.

6. **Conjunction Elimination**

   Similarly, we have seen that $\vDash (s_1 \wedge s_2) \supset s_1$ as well as $\vDash (s_1 \wedge s_2) \supset s_2$. So Conjunction Elimination is valid.

7. **Conjunction Introduction**

   This is a bit more complicated but not by much. We know that

   $$\vDash s_1 \supset \big[s_2 \supset \big(s_1 \wedge s_2\big)\big].$$

   So

   $$\text{If } \Gamma \vDash s_1, \text{ then } \Gamma \vDash s_2 \supset (s_1 \wedge s_2).$$

   And this means that if additionally $\Delta \vDash s_2$, then $\Gamma, \Delta \vDash s_1 \wedge s_2$. Put together, we have:

   $$\text{If } \Gamma \vDash s_1 \text{ and } \Delta \vDash s_2, \text{ then } \Gamma, \Delta \vDash s_1 \wedge s_2.$$

   I.e., if $\Gamma \vdash s_1$ and $\Delta \vdash s_2$ are correct, then $\Gamma, \Delta \vdash s_1 \wedge s_2$ is correct. Thus, Conjunction Introduction is valid.

8. **Disjunction Elimination**

   Let's leave this as an exercise.

9. **Negation Introduction**

   Let's leave this as an exercise, too.

4.7.3 Drawing the conclusion

Since all the inference rules are valid, given any series of *n* correct sequents, inferring in accordance with our proof system will result in a series of *n+1* correct sequents. Since we know that a series of 1 sequent is a series with only correct sequents—remember, only Assumption Introduction can get that single line in place—we now know that extending such a series to 2 sequents will leave us with a series of 2 correct sequents, and ditto if we continue to infer a third sequent, fourth sequent, etc. It does not matter how long the series of sequents gets, if each sequent was inferred using one or the other of our rules of inference, then the series of sequents contains only correct sequents. Therefore, any sentence proven to be a theorem is indeed a tautology. Our system of sentential logic is sound.

4.8 Exercise: Completeness

This section walks you through an argument showing Completeness: any tautology of $\mathcal{L}$ is also a theorem (if $\vDash t$, then $\vdash t$).

The questions give an outline of the proof of Completeness ('proof' in the colloquial sense—it's not a proof in the technical sense we introduced) in the style of the proof of Completeness given by László Kalmár (1935). Your task is to fill in the details. Just as in the fill-in-the-blanks exercises, it makes sense to look at the whole thing first—it might be that some later parts give you clues about how to do earlier parts. And if you don't know how to do a question, you can proceed under the pretense that you managed to answer the question.

The easiest way to understand how our argument works is by thinking of what the truth table of a sentence s tells us. A truth tables lists all interpretations and it tells us for each interpretation whether or not s is true in that interpretation. Let us think of a interpretation not as a series of *T*s and *F*s but as a set of sentences: for each atomic sentence a_i, if the table says *T*, a_i is in the interpretation, and if the table says *F*, $\neg a_1$ is in the interpretation; and no other sentences are in the interpretation. For instance, for $\mathcal{L}$ with two atomic sentences a_1 and a_2, the interpretations are:

1. a_1, a_2 ..... corresponds to *TT*
2. $\neg a_1, a_2$ ..... corresponds to *FT*
3. $a_1, \neg a_2$ ..... corresponds to *TF*
4. $\neg a_1, \neg a_2$ ..... corresponds to *FF*

In the following, we will be using this understanding of interpretations.

The Proof

1. Let's first take some sentence t and see what happens if $\mathcal{M} \vdash t$ for any $\mathcal{M}$. We will show that in that case $\vdash t$. (Note: $\mathcal{M}$ is a placeholder for a set of sentence. So it must not appear on the succedent side of a sequent.)

   Suppose t is a sentence constructed using n atomic sentences a_1 through a_n, and further suppose that $\mathcal{M} \vdash t$ for any interpretation $\mathcal{M}$. That is, we are supposing that the following 2^n sequents can all be proven (we list them in the order we list interpretations in a truth table):

   1. $a_1, \quad a_2, \quad a_3, \quad ..., \quad a_n \quad \vdash \quad t$
   2. $\neg a_1, \quad a_2, \quad a_3, \quad ..., \quad a_n \quad \vdash \quad t$
   3. $a_1, \quad \neg a_2, \quad a_3, \quad ..., \quad a_n \quad \vdash \quad t$
   4. $\neg a_1, \quad \neg a_2, \quad a_3, \quad ..., \quad a_n \quad \vdash \quad t$
   5. $a_1, \quad a_2, \quad \neg a_3, \quad ..., \quad a_n \quad \vdash \quad t$
   6. $\neg a_1, \quad a_2, \quad \neg a_3, \quad ..., \quad a_n \quad \vdash \quad t$
   7. $a_1, \quad \neg a_2, \quad \neg a_3, \quad ..., \quad a_n \quad \vdash \quad t$
   8. $\neg a_1, \quad \neg a_2, \quad \neg a_3, \quad ..., \quad a_n \quad \vdash \quad t$

   $\vdots$

   2^n. $\neg a_1, \quad \neg a_2, \quad \neg a_3, \quad ..., \quad \neg a_n \quad \vdash \quad t$

   (a) From the first pair of sequents in the list above, derive:

   $a_2, \quad a_3, \quad ..., \quad a_n \quad \vdash \quad t$

   You may appeal to exactly one theorem that we have named earlier, and you may appeal to this one theorem in the subsequent problems; but you may not appeal to any other theorem.

   (b) From the second pair of sequents, derive:

   $\neg a_2, \quad a_3, \quad ..., \quad a_n \quad \vdash \quad t$

(c) From the results of (a) and (b), derive:

$a_3, \quad ..., \quad a_n \quad \vdash \quad t$

(d) From the sequents number 5 through 8 in the above list, derive:

$\neg a_3, \quad ..., \quad a_n \quad \vdash \quad t$

(e) Notice how the datum side of the sequents get shorter? Explain why you eventually will be able to derive the following two sequents:

$a_n \quad \vdash \quad t$

$\neg a_n \quad \vdash \quad t$

(f) Derive $\vdash t$ from the result of (e).

This results show that if $\mathcal{M} \vdash t$ for every interpretation $\mathcal{M}$, then $\vdash t$ (i.e., t is a theorem). This is one important piece of the puzzle.

2. Let's move to another piece of the puzzle. We will show that for any sentence s and interpretation $\mathcal{M}$, either we can prove that $\mathcal{M} \vdash s$ or we can prove that $\mathcal{M} \vdash \neg s$. We will make use of the classification of sentences discussed in Chapter 2.8.

   (a) First, let p be a class-0 sentence. Show that for any interpretation $\mathcal{M}$, either we can *prove* $\mathcal{M} \vdash p$ or we can *prove* $\mathcal{M} \vdash \neg p$. (Hint: notice that $\mathcal{M}$ must either contain p or contain $\neg p$.)

   (b) Secondly, suppose that for any sentence p of class-n either $\mathcal{M} \vdash p$ or $\mathcal{M} \vdash \neg p$. All the subquestions of this problem are under this supposition. Let us show that on this supposition, it is also true that for any sentence s of class-(n+1) either $\mathcal{M} \vdash s$ or $\mathcal{M} \vdash \neg s$.

   We can show this by working through all the possible cases.

   Construct each of the following indicated derivations—except x, xi, xii for which you should fill in the blanks. Some derivations

are trivial; others require a bit more work; and not all derivations will make use of all the indicated premises. Most importantly, you can find clues about how to do most of these from previous exercises/practice material/tests (or even the current exercise):

i. From $\mathcal{M} \vdash p$ and $\mathcal{M} \vdash q$, derive $\mathcal{M} \vdash p \wedge q$.

ii. From $\mathcal{M} \vdash p$ and $\mathcal{M} \vdash q$, derive $\mathcal{M} \vdash p \vee q$.

iii. From $\mathcal{M} \vdash p$ and $\mathcal{M} \vdash q$, derive $\mathcal{M} \vdash p \supset q$.

iv. From $\mathcal{M} \vdash \neg p$ and $\mathcal{M} \vdash q$, derive $\mathcal{M} \vdash \neg(p \wedge q)$.

v. From $\mathcal{M} \vdash \neg p$ and $\mathcal{M} \vdash q$, derive $\mathcal{M} \vdash p \vee q$.

vi. From $\mathcal{M} \vdash \neg p$ and $\mathcal{M} \vdash q$, derive $\mathcal{M} \vdash p \supset q$.

vii. From $\mathcal{M} \vdash p$ and $\mathcal{M} \vdash \neg q$, derive $\mathcal{M} \vdash \neg(p \wedge q)$.

viii. From $\mathcal{M} \vdash p$ and $\mathcal{M} \vdash \neg q$, derive $\mathcal{M} \vdash p \vee q$.

ix. From $\mathcal{M} \vdash p$ and $\mathcal{M} \vdash \neg q$, derive $\mathcal{M} \vdash \neg(p \supset q)$.

x. Fill in missing items. This derives from $\mathcal{M} \vdash \neg p$ and $\mathcal{M} \vdash \neg q$ to $\mathcal{M} \vdash \neg(p \wedge q)$.

1.	$\mathcal{M}$	$\vdash \neg p$	..... premise
2.	$\mathcal{M}$	$\vdash \neg q$	..... premise
3.	$p \wedge q$	$\vdash p \wedge q$	..... A
4.	__	$\vdash$ __	..... 3,__
5.	$\mathcal{M}$, __	$\vdash$ __	 1
6.	$\mathcal{M}$	$\vdash \neg(p \wedge q)$	..... 4,5,¬I

xi. Fill in missing items. This derives from $\mathcal{M} \vdash \neg p$ and $\mathcal{M} \vdash \neg q$ to $\mathcal{M} \vdash \neg(p \vee q)$.

1.	$\mathcal{M}$	$\vdash \neg p$	..... premise
2.	$\mathcal{M}$	$\vdash \neg q$	..... premise
3.	$p \vee q$	$\vdash p \vee q$	..... __
4.	p	$\vdash p$	..... __
5.	$p, p \vee q$	$\vdash p$	..... __
6.	$\mathcal{M}, p \vee q$	$\vdash \neg p$	..... __
7.	$\mathcal{M}, p$	$\vdash \neg(p \vee q)$	..... __
8.	q	$\vdash q$	..... __
9.	$q, p \vee q$	$\vdash q$	..... __
10.	$\mathcal{M}, p \vee q$	$\vdash \neg q$	..... __
11.	$\mathcal{M}, q$	$\vdash \neg(p \vee q)$	..... __
12.	$\mathcal{M}, p \vee q$	$\vdash \neg(p \vee q)$	..... __
13.	$\mathcal{M}$	$\vdash \neg(p \vee q)$	..... __

xii. Fill in missing items. This derives from $\mathcal{M} \vdash \neg p$ and $\mathcal{M} \vdash \neg q$ to $\mathcal{M} \vdash p \supset q$.

1.	$\mathcal{M}$	$\vdash \neg p$	..... premise
2.	$\mathcal{M}$	$\vdash \neg q$	..... premise

3.	p	$\vdash p$	..... A
4.	__	$\vdash$ __	 1
5.	__	$\vdash$ __	 3
6.	$\mathcal{M}, p$	$\vdash$ __	..... 4,5,¬I
7.	$\mathcal{M}, p$	$\vdash$ __	..... 6,¬E
8.	$\mathcal{M}$	$\vdash p \supset q$	..... 7,⊃I

xiii. From $\mathcal{M} \vdash p$, derive $\mathcal{M} \vdash \neg\neg p$.

(c) Let's make sure you see the point of the above:

i. Explain why if s is a conjunction of two class-n sentences, then either $\mathcal{M} \vdash s$ or $\mathcal{M} \vdash \neg s$.

ii. Explain why if s is a disjunction of two class-n sentences, then either $\mathcal{M} \vdash s$ or $\mathcal{M} \vdash \neg s$.

iii. Explain why if s is a conditional of two class-n sentences, then either $\mathcal{M} \vdash s$ or $\mathcal{M} \vdash \neg s$.

iv. Explain why if s is a negation of a class-n sentence, then either $\mathcal{M} \vdash s$ or $\mathcal{M} \vdash \neg s$.

(d) Explain why it follows that:
If for any sentence p of class-n either $\mathcal{M} \vdash p$ or $\mathcal{M} \vdash \neg p$, then any sentence s of class-(n+1) is such that either $\mathcal{M} \vdash s$ or $\mathcal{M} \vdash \neg s$.

3. We now assemble our pieces of the puzzle.

(a) Explain why if t is a tautology, it is not the case that $\mathcal{M} \vDash \neg t$ for any interpretation $\mathcal{M}$.

(b) Explain why if $\mathcal{M} \vdash \neg t$, then $\mathcal{M} \vDash \neg t$ (Hint: recall we already know an important feature of our proof system).

(c) Explain why if $\vDash t$, then $\mathcal{M} \vdash t$ for any interpretation $\mathcal{M}$.

4. Explain why it follows from what we have shown that our proof system is complete.

Q.E.D. (*quod erat demonstandum.* 'This is what was to be shown' in Latin.) Congratulations! You are done.

Note

The above argument also gives you a recipe for proving any tautology by mimicking the construction of truth tables. For example, here is a proof of $\vdash (p \wedge q) \supset p$ suggested by the argument:

1. $p \qquad \vdash p$ ..... A

2.	q	$\vdash q$	A
3.	$\neg p$	$\vdash \neg p$	A
4.	$\neg q$	$\vdash \neg q$	A
5.	p, q	$\vdash p$	1
6.	p, q	$\vdash q$	2
7.	$\neg p, q$	$\vdash \neg p$	3
8.	$\neg p, q$	$\vdash q$	2
9.	$p, \neg q$	$\vdash p$	1
10.	$p, \neg q$	$\vdash \neg q$	4
11.	$\neg p, \neg q$	$\vdash \neg p$	3
12.	$\neg p, \neg q$	$\vdash \neg q$	4
13.	p, q	$\vdash p \wedge q$	5,6,$\wedge$I
14.	$p \wedge q$	$\vdash p \wedge q$	A
15.	$p \wedge q$	$\vdash p$	14,$\wedge$E
16.	$\neg p, q, p \wedge q$	$\vdash \neg p$	7
17.	$\neg p, q$	$\vdash \neg(p \wedge q)$	15,16,$\neg$I
18.	$p \wedge q$	$\vdash q$	14,$\wedge$E
19.	$p, \neg q, p \wedge q$	$\vdash \neg q$	10
20.	$p, \neg q$	$\vdash \neg(p \wedge q)$	18,19,$\neg$I
21.	$\neg p, \neg q, p \wedge q$	$\vdash \neg p$	11
22.	$\neg p, \neg q$	$\vdash \neg(p \wedge q)$	15,21,$\neg$I
23.	$p, q, p \wedge q$	$\vdash p$	5
24.	p, q	$\vdash (p \wedge q) \supset p$	23,$\supset$I
25.	$\neg p, q, \neg p$	$\vdash \neg p$	7
26.	$p \wedge q, \neg p$	$\vdash p$	15
27.	$\neg p, q, p \wedge q$	$\vdash \neg\neg p$	25,26,$\neg$I
28.	$\neg p, q, p \wedge q$	$\vdash p$	27,$\neg$E
29.	$\neg p, q$	$\vdash (p \wedge q) \supset p$	28,$\supset$I
30.	$p, \neg q, p \wedge q$	$\vdash p$	9
31.	$p, \neg q$	$\vdash (p \wedge q) \supset p$	30,$\supset$I
32.	$\neg p, \neg q, \neg p$	$\vdash \neg p$	11
33.	$\neg p, \neg q, p \wedge q$	$\vdash \neg\neg p$	26,32,$\neg$I
34.	$\neg p, \neg q, p \wedge q$	$\vdash p$	33,$\neg$E
35.	$\neg p, \neg q$	$\vdash (p \wedge q) \supset p$	34,$\supset$I
36.		$\vdash p \vee \neg p$	EM
37.	q	$\vdash (p \wedge q) \supset p$	24,29,36,$\vee$E
38.	$\neg q$	$\vdash (p \wedge q) \supset p$	31,35,36,$\vee$E
39.		$\vdash q \vee \neg q$	EM
40.		$\vdash (p \wedge q) \supset p$	37,38,39,$\vee$E

You see why no one teaches this recipe for generating proofs. It's spending 40 lines to prove something that can be proved in three lines. But it shows a point of principle. It is not too difficult to write computer code to generate such a proof for any theorem no matter how complex the theorem.

4.9 Significance of Soundness and Completeness

What is the point of proving soundness and completeness? We started our discussion of the proof system by considering ordinary language arguments and how they might be formalized. Our formal system is designed to keep track of what supports what. Soundness of our system means that if we start with some claims conclusively supported by the evidence (the datums of the premises are the evidence, the succedents the claims), then inferences drawn in accordance with our proof system will correctly identify what else is supported by that evidence. Our proof system is immune to fallacious inferences.

Secondly, since our proof system is complete, it is possible to infer to everything that is conclusively supported by the evidence. There is no need to worry that reasoning in accordance with the proof system will make us miss out on some things supported by the evidence.

Taken together, this means that our proof system is just right. We can reach all conclusions we want to reach, while keeping us from drawing conclusions that are illicit.

Our proof system is extracted from observations of what we regard as good reasoning. What we have shown is that what we regard as good cases of slow and careful reasoning really are good. While there are some important limitations of our proof system that we have noted earlier—and we will see more very soon—completeness shows that what we can do, we can do well. And that is a good thing to know about ourselves.

The arguments for soundness and completeness are not arguments formulated within our proof system. They are theorems *about* our proof system and are not themselves theorems of the proof system. They are known as *meta-theorems*. Meta-theorems are proven in the meta-language. That is why, for instance, many 'if, then' statements used in the course of the argument did not use the horseshoe symbol (⊃)—our meta-language is English, and the horseshoe symbol is an expression of $\mathcal{L}$ but not of English.

Chapter 5

Predicate Logic

5.1 Introduction

In the previous chapters we have seen sentential logic. It is called 'sentential' because it focuses on sentences and logical relations that hold between sentences. We have seen that the proof system we constructed is complete. Any tautology can be proven and that means that whenever the truth of s guarantees the truth of t—i.e. $s \vDash t$—we can also prove that $\vdash s \supset t$. And this means that any valid argument can be turned into a successful derivation. You might therefore hope that we are done with formalizing reasoning. But, alas, things are more complicated.

Consider the following piece of reasoning:

> **Example 5.1.1.** All whales are mammals. Ed is a whale. So Ed is a mammal.

This seems like a fine piece of reasoning. On the assumption that all whales are mammals and that Ed is a whale, it surely follows that Ed is a mammal. But how would we formalize this?

Let's use the following keys:

A: All whales are mammals.
B: Ed is a whale.
C: Ed is a mammal.

Then the above reasoning looks as follows:

1. $\Gamma \quad \vdash A \quad$ ..... premise
2. $\Delta \quad \vdash B \quad$ ..... premise
3. $\Gamma, \Delta \quad \vdash C \quad$ ..... 1,2,??

There is no inference rule that would license the move to line 3 from lines 1 and 2: It cannot be that given some arbitrary three sentences s_1, s_2 and s_3, we can infer from $\Gamma \vdash s_1$ and $\Delta \vdash s_2$ to $\Gamma, \Delta \vdash s_3$. But the argument about Ed in the example above is surely cogent. So we have an example of an argument that can be stated in plain English without difficulties, but cannot be formalized in sentential logic.

Let's think about the example a bit more. Compare:

- All whales are mammals.
- If anything is a whale, then it is a mammal.

These two sentences say the same thing: the first is true iff. the second is true. Let's use this and rewrite the example:

> **Example 5.1.2.** All whales are mammals. That is to say, if anything is a whale, then it is a mammal. So if Ed is a whale, then Ed is a mammal. Ed is a whale. It follows that Ed is a mammal.

Breaking this up into lines, we get:

4. If anything is a whale, then it is a mammal.

$\therefore$ 5. If Ed is a whale, then Ed is a mammal.

6. Ed is a whale.

$\therefore$ 7. Ed is a mammal.

Lines 5 through 7 pose no difficulties for sentential logic. We can partially formalize the argument as (using the same keys as before):

8. ???

9. $\Delta \quad \vdash B \supset C \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$ 8,?

10. $\Theta \quad \vdash B \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$ premise

11. $\Delta, \Theta \quad \vdash C \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$ 9,10,$\supset$E

The trouble is with formalizing the move from 4 to 5. How do we get to line 9 (which corresponds to line 5 of the version in English)?

You might be tempted by the thought that 4 is a conditional connecting the sentences 'anything is a whale' with 'it is a mammal'. So let's try the following keys:

D: Anything is a whale.
E: It is a mammal.

And then we get:

8. Γ	$\vdash D \supset E$	..... premise
9. Δ	$\vdash B \supset C$	..... 8,?
10. Θ	$\vdash B$	..... premise
11. Δ, Θ	$\vdash C$	..... 9,10,⊃E

But this will not do. First of all, there is no way to derive 9 from 8. Secondly, and much more importantly for our purposes, 4 (if anything is a whale, then it is a mammal) is *not* a conditional. If it were, then both 'anything is a whale' and 'it is a mammal' must be sentences. Take the former. That might be interpreted to mean that everything is a whale. That's false. Given the truth conditions of a conditional, the whole conditional would therefore be true no matter what the consequent. For instance, it would be true that if anything is a whale, it is a reptile. And that can't be right. Moreover, take 'it is a mammal'. What does the 'it' in 'it is a mammal' refer to? There is nothing in particular the 'it' refers to. And that means that 'it is a mammal' cannot be true or false: there is nothing of which it is claimed that *it* is a mammal. So it is not a sentence, and 8 misconstrues 4.

Let's think again. There should be no doubt that 5 follows from 4. When you compare the two, you will notice that they have more in common than the 'if, then' construction. They are both instances of:

if _____ is a whale, then _____ is a whale.

The difference between 4 and 5 consists in how the blanks are filled in. This is a similarity that cannot be captured with sentential logic because the language of sentential logic is incapable of revealing structures smaller than whole sentences. But it seems that the cogency of the move from 4 to 5 has something to do with these sub-sentential structures.

In the following we will introduce a formal language—the language of predicate logic—that is capable of representing sub-sentential structures and formulate rules of inferences that can exploit them. Taken together, we will have a system of *first-order predicate logic*.

5.2 Quantifiers and Variables

Take a look again at

if _____is a whale, then _____is a mammal

This is not a sentence. It is more like a *template* of a sentence. You can generate a sentence by filling in the blanks. Let us use letters like x and y to represent the blanks. Using letters allows us to distinguish between templates like:

- if x is a whale, then x is a mammal
- if x is a whale, then y is a mammal

The rule is that blanks represented by the same letter within a template must be filled in the same way. So in the first template, if you plug 'Ed' into the first x, you must plug 'Ed' into the second as well. But blanks represented by different letters can be filled in differently. For instance, in the second template you can fill 'Ed' into the x and 'Gina' into y and get the sentence 'if Ed is a whale, then Gina is a mammal' which is a sentence you cannot get by filling in the blanks of the first template. We will call letters that represent blanks *variables*.

'If Ed is a whale, then Ed is a mammal' and 'if Ed is a whale, then Gina is a mammal' are both sentences. Why? Because each of them makes a claim about the world and can be true or false. A template does not yet make any claims about the world and that is why the templates above are not sentences. We can get sentences by plugging into the variables.

But there is another way of getting a sentence out of the templates. Consider prefixing a template with 'for any x':

for any x, if x is a whale, then x is a mammal

In more ordinary English, we might put this as: everything is such that if it is a whale, then it is a mammal. Or more colloquially: if anything is a whale, it is a mammal. Even more colloquially: all whales are mammals. This, too, is a sentence since it is the sort of thing that is either true or false. And this is not a complex sentence formed by connecting two sentences. Rather, it is a sentence whose parts are not themselves sentences.

There is one more way of getting a sentence out of a template that we will be discussing in this chapter. We can prefix the template above with 'there is an x such that' and get a sentence as well:

there is an x such that if x is a whale, then x is a mammal

The language of predicate logic formalizes these features of sentence templates.

Before we proceed, let me make some more observations. The template like 'if x is whale, then x is a mammal' is itself made out of two templates:

x is a whale

x is a mammal

Connecting these templates with the 'if ... then ...' construction gives us the template 'if x is a whale, then x is a mammal'. But we could easily connect these two with a different connective. For example, 'x is a whale and x is a mammal', or 'x is a whale or x is a mammal', etc. So templates can be turned into compound templates using the familiar logical connectives.

While some templates are compound templates that connect two or more templates, others like 'x is a whale' are not. These atomic (i.e., non-compound) templates take subject-verb form in English. We will have something similar in our formal language.

Not all sentences of English take a simple subject-verb form. In fact, most don't. For instance, 'Hermione is married to Ron' involves Hermione and Ron, not just Hermione. And 'Harry mediates between Hermione and Ron' involves Harry, Hermione and Ron. A template for generating such a sentence would be 'x mediates between y and z'. We want our formal language to have such templates as well.

5.3 Formalization

Let us introduce a formal language $\mathcal{L}_Q$. The Q stands for quantifier. Just as with $\mathcal{L}$, our aim is to construct a bare-bones language that is simple and does the job of representing what we want to represent but nothing else.

Let us start. $\mathcal{L}_Q$ has *formulas*. The intuitive characterization of a formula is that both sentences and templates are formulas. The formulas are such that:

- If φ is a formula, so is $\neg\varphi$.
- If φ and ψ are formulas, so are $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \supset \psi)$.

(φ is Greek lowercase phi, and ψ is Greek lowercase psi). Notice that if we restrict φ and ψ to sentences, this is the same we had for $\mathcal{L}$. So we can expect our new language $\mathcal{L}_Q$ to carry over many of the features of $\mathcal{L}$.

What does a formula look like? We need a general characterization that can cover both sentences and templates. Compare the sentence 'Ed is a whale' with the template 'x is a whale'. They have something in common, namely the 'is a whale' part. But they differ in that 'Ed' is a proper name whereas x is a variable.

- $\mathcal{L}_Q$ has *predicates* corresponding to expressions like 'is a whale' in English.
- $\mathcal{L}_Q$ has *variables* like the x and y that we have been using.
- $\mathcal{L}_Q$ has *constants* which correspond to proper names likes 'Ed' in English.

If we combine a predicate with a variable, we get a template. If we combine a predicate with a constant, we get a sentence. Let us introduce the word *term* to cover both variables and constants. Then a formula is something that combines a predicate with a term.

Some predicates must be combined with multiple terms to generate a formula. For instance, the predicate corresponding to '... mediates between ... and ...' combines with three terms to yield a formula; e.g., 'x mediates between y and z'. A predicate that combines with n terms to yield a formula is called an n-place predicate. So '... mediates between ... and ...' is a 3-place predicate. We will say the following about formulas in $\mathcal{L}_Q$:

- If $\mathcal{F}$ is an n-place predicate, τ_1, τ_2, ..., τ_n are terms, then $\mathcal{F}\tau_1\tau_2\ldots\tau_n$ is a formula.

(τ is the Greek lowercase tau). So we give $\mathcal{L}_Q$ a bunch of predicates, a bunch of constants, and a bunch variables, and that gives us a bunch of formulas (in fact, infinitely many). You can read, for example, Fx as 'x is F' and Fc as 'c is F'. In this course, we will mostly be dealing with 1-place, or *monadic*, predicates.

This looks scarier than it is. Notice that once we have formulas, they combine to form more complex formulas according to the same principles as the compound sentences of $\mathcal{L}$ we saw earlier. So what is new is a way of representing the internal structure of sentences and templates. Think of $\mathcal{L}_Q$ as a language with a very primitive grammar: any atomic formula starts with a predicate and has one or more terms attached. Because the grammar is extremely primitive, the order of the terms is important (just as in English, word order matters).

Some examples:

- Let W be a 1-place predicate that can be translated into 'is a whale' in English, and let x be a variable. Then Wx is a formula of $\mathcal{L}_Q$ which can be translated as 'x is a whale' (so it's a template, rather than a sentence). Let e be the name for Ed in $\mathcal{L}_Q$ —i.e., e is a constant of $\mathcal{L}_Q$. Then We is a formula of $\mathcal{L}_Q$ that we can translate as 'Ed is a whale' (so it's a sentence).

- Let M be a 3-place predicate M such that $Mxyz$ translates into 'x mediates between y and z' (it's often easier to explain what a predicate in $\mathcal{L}_Q$ means by providing a full template for a sentence). if h_1 is a constant that names Harry, h_2 a constant that names Herminone, and r a constant that names Ron, then Mh_1h_2r can be translated as 'Harry mediates between Herminone and Ron'.

- Let T be a 2-place predicate such that Txy translates into 'x is taller than y'. Then if s is a constant naming Sherlock and j is a constant naming John, Tsj can be translated as 'Sherlock is taller than John' while Tjs can be translated into 'John is taller than Sherlock'. Word order matters.

There is one more important feature of $\mathcal{L}_Q$. Recall that we can take a template

like 'x is a whale', prefix it with 'for any x' or 'there is an x such that' and get a sentence out of it. To accomplish the same in $\mathcal{L}_Q$, we will say:

- if φ is a formula and υ a variable, $\forall\upsilon\varphi$ is a formula.

(υ is Greek lowercase upsilon). You can read $\forall\upsilon$ as 'for all υ' and it is known as the *universal quantifier*. For instance , $\forall xWx$ means that for all x, x is a whale (assuming Wx means that x is a whale)—or more colloquially, everything is a whale. And if Mx means that x is a mammal, then the sentence 'for all x, if x is a whale, then x is a mammal' can be put as $\forall x(Wx \supset Mx)$.

We will also have another quantifier called the existential quantifier:

- if φ is a formula and υ a variable, $\exists\upsilon\varphi$ is a formula.

You can read $\exists\upsilon$ as 'there is an υ such that'. For instance, $\exists xWx$ means 'there is an x such that x is a whale' (again keeping the same meaning for W as above)—or, more colloquially, 'there are whales' (the plural 'whales' is not meant to commit us to the existence of multiple whales; there might be just one).

Altogether, we can characterize a language $\mathcal{L}_Q$—a language of first-order predicate logic, as it is known—in the following way:

1. There are infinitely but countably many constants.
2. There are infinitely but countably many variables.
3. There are infinitely but countably many predicates.
4. If $\mathcal{F}$ is an *n*-place predicate, τ_1, τ_2, ..., τ_n are terms (constants and variables), then $\mathcal{F}\tau_1\tau_2\ldots\tau_n$ is a formula.
5. If φ is a formula, then $\neg\varphi$ is a formula.
6. If φ and ψ are formulas, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \supset \psi)$ are all formulas.
7. If φ is a formula and υ a variable, then $\forall\upsilon\varphi$ is a formula.
8. if φ is a formula and υ a variable, $\exists\upsilon\varphi$ is a formula.
9. Nothing else is a formula.

It is customary (but not required) to use lower case letters from the end of the Roman alphabet like x, y, z as variables. And it is customary to use upper case Roman alphabet letters like F, G, H for predicates, and lower case letters from the start of the Roman alphabet like a, b, c for constants. If we need more, we will generally resort to subscripts. And for ease of exposition, we will call the quantifiers *connectives* so we can say that the main connective of $\forall x F x$ is $\forall x$.

It is good practice to always explicitly state which letters are used as variables, which as predicates, and which as constants. And no letter should be used as both a variable and a constant in the same context.

Notice that a formula of $\mathcal{L}_Q$ need not be a sentence. We need to do a little bit more work to be able to clearly specify what a sentence of $\mathcal{L}_Q$ is.

5.4 Scope, Bound and Free Variables

Let Lxy stand for 'x loves y'. We can prefix this with a universal quantifier:

1. $\forall xLxy$

This means 'For any x, x loves y'. The universal quantifier does not affect y. y is still a simple blank so that 1 is still a template. We say that the x in Lxy is *bound* by the quantifier $\forall x$. y is not bound by any quantifier. We say that y is a *free* variable—y is still just a blank and you are free to plug into it a constant to get a sentence, but you cannot plug a constant into x given that it is bound.

We can prefix another quantifier to bind y. E.g.:

2. $\forall y\forall xLxy$

For any y and any x, x loves y. That is, we can plug anything into y and plug anything into x and the result will be true. And this means that in this case, the order of the quantifiers does not matter:

3. $\forall x\forall yLxy$

3 means the same as 2. But this is a special case. Usually, the order of quantifiers matters greatly. Start with 1 again but let's think of a different way of binding y:

4. $\exists y\forall xLxy$

There is a y, such that any x loves y. So if everybody loves Raymond, 4 is true. But if there isn't anyone who is loved by everyone, 4 would be false. Now consider the following:

5. $\forall x\exists yLxy$

For any x, there is a y such that x loves y. Notice that this does not require that there be someone whom everyone loves. Maybe no one loves Raymond. But that does not conflict with everyone's having someone they love (it's just

that that someone is never Raymond). 5 would still be true in that case, but not 4. 4 and 5 mean different things.

To capture the difference between 4 and 5, logicians speak of the *scope* of a quantifier. The scope of a quantifier is simply the formula to which it is attached. The scope of the existential quantifier $\exists y$ in 4 is the formula $\forall x Lxy$. But in 5 it is Lxy. The scope of the universal quantifier $\forall x$ in 4 is Lxy but in 5 it is $\exists y Lxy$. Differences in scope usually matter a great deal.

A quantifier always names the variable it binds. The quantifier $\forall \upsilon$ binds all instances of υ that would be free without the quantifier. For instance, in 2 above, the quantifier $\forall y$ binds y in $\forall x Lxy$ because the y in that formula is still free. On the other hand, consider:

6. $\forall x \forall x Lxy$

In 6, the left-most quantifier $\forall x$ does not bind anything because it names the variable to be bound as x but there is no free x within its scope. And the y remains free in 6. It is in general bad practice to attach a quantifier that does not bind anything even though the rules for $\mathcal{L}_Q$ do not forbid it.

There are some tricky issues to be aware of when dealing with scope. Take the following two templates:

7. Wx
8. Mx

(Wx means x is a whale, Mx means x is a mammal). We can take 7 and prefix it with a quantifier:

9. $\exists x Wx$

We can take 9 and 8 and connect them with the conditional to get a new formula:

10. $\exists x Wx \supset Mx$

Notice that the x in Mx is still free because it is not within the scope of the existential quantifier at the left end. This means we can bind the x in Mx. For example:

11. $\forall x(\exists x Wx \supset Mx)$

Notice that the universal quantifier binds the x in Mx but does not bind the x in Wx because the latter x is already bound by the existential quantifier. While 11 is allowed by the rules of $\mathcal{L}_Q$ it is confusing and it will make life really difficult when we move to inferences involving quantifiers. The solution is to rename one of the x's. They are just blanks with labels to keep track of what gets plugged into what. So there is no change in meaning if we rename one of the x's. E.g.:

12. $\forall y(\exists x Wx \supset My)$

For any y, if there are whales, then y is a mammal (that's probably false).

Going forward, we will require that all variables of a given name within the scope of a quantifier must be bound by the same quantifier. 11 fails this requirement because not all x's within the scope of the universal quantifier are bound by it. But 12 meets this requirement.

One more point of notation. Given the variable υ, we will write $\varphi(\upsilon)$ to stand for a formula in which υ occurs as a free variable. φ can be any formula so it may contain other variable that are already bound by quantifiers within φ. But when we write $\varphi(\upsilon)$, υ is free within φ. So when we write $\forall x\varphi(x)$, the universal quantifiers binds the x in φ because x is free within φ. When there are multiple free variables in φ, we will indicate that by a comma-separated list; e.g., $\varphi(\upsilon_1, \upsilon_2)$ is a formula in which there are two free variables υ_1 and υ_2.

We will also sometimes write $\varphi(\kappa)$ where κ is a constant to mean that some free variable in φ was replaced by the constant κ. As I said earlier, it is very good practice to always be explicit about which letters are used for constants, and which for variables.

5.4.1 Sentences of $\mathcal{L}_Q$

We now can specify which formulas of $\mathcal{L}_Q$ are sentences: a formula $\mathcal{L}_Q$ is a sentence iff. it contains no free variables.

5.5 Semantics for $\mathcal{L}_Q$

So far we have relied on an intuitive understanding of what sentences of $\mathcal{L}_Q$ mean. It is time to introduce a more formal account of the semantics of $\mathcal{L}_Q$. In particular, we want to have a more rigorous definition of when a sentence of $\mathcal{L}_Q$ is true.

In the case of sentential logic, we used interpretations to figure out things like whether a given sentence is a tautology. An interpretation there assigned truth values to atomic sentences, and that enabled us to figure out the truth value of a compound sentence in that interpretation. We will do something similar for our language of predicate logic. However, since the smallest building blocks of the sentences of predicate logic are not entire sentences, but terms and predicates, our interpretations make assignments to those. Since the building blocks are not sentences, the interpretation cannot assign truth values to them. So what does an interpretation do? The following specifies what an interpretation does.

First, an interpretation of $\mathcal{L}_Q$ specifies a *domain of discourse*, or just *domain*. This is the set of things our language $\mathcal{L}_Q$ can talk about. 'Thing' is understood in the broadest possible sense: anything that can be named can be a thing. There is no requirement of any kind of commonality among the members of the domain save that they all must be 'things.' And a domain of discourse can have finitely many, countably many, or even uncountably many things as members. The set {Julius Caesar, Beethoven's Ninth Symphony, the United Nations Security Council, Cantor's Diagonal Argument} can be a domain of discourse, and so can be the set of all rational numbers. However, the domain of discourse cannot be empty.

Next, an interpretation assigns to each constant of $\mathcal{L}_Q$ a member of the domain of discourse. That is, it specifies for each constant a thing in the domain that is named—or *referred to*—by that constant. Let's call the thing named by a constant its *referent*. Notice that different constants can name the same thing: some things have more than one name like the first emperor of Rome who is variously known as Octavius, Octavian, Augustus. But there is no constant without a referent; i.e., every name is a name of something. Finally, not everything in the domain need be assigned as the referent of a constant—like most streets in Japan, some things might have no names.

Thirdly, an interpretation assigns to each predicate of $\mathcal{L}_Q$ an *extension*. For a one-place predicate $\mathcal{F}$, the interpretation assigns a set of members of the

domain. Intuitively, the extension of $\mathcal{F}$ is the set of all the members of the domain that are $\mathcal{F}$. For example, if the domain is the set of all students at the Claremont Colleges, an interpretation could assign all the Pomona College students to P, which would enable us to translate Px as 'x is a Pomona College student'. For a two-place predicate R, the extension is a set of ordered pairs. For instance, the interpretation might assign to H all the pairs $< \mathfrak{m}, \mathfrak{n} >$ such that $\mathfrak{m}$ is taller than $\mathfrak{n}$ (those are m and n in a font style known as Fraktur). That would enable us to translate Hxy as 'x is taller than y'. Notice that the pairs must be ordered, i.e., the order in which the members of the pairs are listed matters. More generally, for each n-place predicate $\mathcal{F}$, the interpretation assigns a set of ordered n-tuplets as the extension of $\mathcal{F}$. Finally, the extension of a predicate may be empty: there might be nothing that is $\mathcal{F}$.

The above suffices to explain what it takes for an atomic sentence of $\mathcal{L}_Q$ that consists of a single 1-place predicate and a constant to be true given an interpretation $\mathfrak{I}$ (that's an I in Fraktur): a sentence $\mathcal{F}\kappa$ is *true in an interpretation* $\mathfrak{I}$ iff. given $\mathfrak{I}$ the referent of κ is in the extension of $\mathcal{F}$. We will also sometimes say that an interpretation $\mathfrak{I}$ *satisfies* $\mathcal{F}\kappa$ true to mean that $\mathcal{F}\kappa$ is true in $\mathfrak{I}$.

More generally, given an n-place predicate $\mathcal{R}$, an interpretation $\mathfrak{I}$ makes true $\mathcal{R}\kappa_1 \ldots \kappa_n$ iff. $< ref(\kappa_1), \ldots, ref(\kappa_n) >$ is in the extension of $\mathcal{R}$ where $ref(\chi)$ is the referent of χ. This takes care of the quantifier-free atomic sentences.

What about quantifier-free compound sentences? We can adapt the understanding from sentential logic:

- $\neg s$ is true in $\mathfrak{I}$ iff. s is not true in $\mathfrak{I}$.
- $s_1 \wedge s_2$ is true in $\mathfrak{I}$ iff. both s_1 and s_1 are true in $\mathfrak{I}$.
- $s_1 \vee s_2$ is true in $\mathfrak{I}$ iff. at least one of s_1 and s_2 is true in $\mathfrak{I}$.
- $s_1 \supset s_2$ is true in $\mathfrak{I}$ iff. at least one of $\neg s_1$ or s_2 is true in $\mathfrak{I}$.

We say that a sentence is *false* iff. it is not true.

Let's move to quantified sentences—those sentences that have a quantifier as the main connective. Consider $\forall x Fx$. We want this to be a reasonable translation of 'everything is F' (that's how we have been treating quantified sentences). The obvious thing to say is: $\forall \upsilon \mathcal{F}\upsilon$ is true in $\mathfrak{I}$ iff. the extension of $\mathcal{F}$ contains every member of the domain. While this is unobjectionable,

it does not help us with figuring out whether $\mathfrak{I}$ makes true $\forall \upsilon(F\upsilon \supset \mathcal{G}\upsilon)$ and other such more complex quantified sentences. Recall that the complex sentence just mentioned is not a compound sentence since it has no parts that are themselves sentences so that we cannot figure out whether it is true in $\mathfrak{I}$ by building up from its component sentences.

Consider $\forall \upsilon \varphi(\upsilon)$. Intuitively, this is true in $\mathfrak{I}$, iff. everything in the domain is as described by φ. So if everything in the domain is the referent of one constant or another, $\forall \upsilon \varphi(\upsilon)$ is true in $\mathfrak{I}$ iff. $\varphi(\kappa)$ is true for all constants of $\mathcal{L}_Q$. But remember that not everything in the domain need be named by a constant. For instance, let the domain be people, and let N be a predicate that has all the nice people in the world in its extension but nobody else. Let's make sure that all our constants refer to one or another person in the extension of N. The not-nice people shall remain nameless. This would make all sentences of the form $N\kappa$ true in the interpretation. Would that mean that everyone is nice? Hardly. What to do?

The solution is to pretend that there is a name for everything. Roughly, $\forall \upsilon \varphi(\upsilon)$ is true in $\mathfrak{I}$ iff. any member $\mathfrak{m}$ of the domain is such that if it had the name κ, $\varphi(\kappa)$ would be true.

Let's be more precise. There are two points to note. First, one way of getting a name for $\mathfrak{m}$ is to take an existing name c and change its interpretation so that it refers to $\mathfrak{m}$. But that risks making a sentence that is true on the old interpretation false on the new one. For instance, if c originally refers to $\mathfrak{t}$ and $\mathfrak{t}$ is in the extension of F, then Fc is true in the original interpretation. But if we change the interpretation of c so that it refers to $\mathfrak{s}$ and $\mathfrak{s}$ is not in the extension of F, Fc would be false on the new interpretation. We do not want to introduce such changes in introducing a constant referring to $\mathfrak{m}$. So what we do is introduce a *new* constant κ and interpret this new constant as referring to $\mathfrak{m}$ while keeping everything else unchanged. Even though $\mathcal{L}_Q$ has infinitely many constants, we can add infinitely many more (e.g., we could start with having constants that have even numbers as subscripts, and then add constants with odd numbers as subscripts as the new ones).

Secondly, we cannot in general name everything all at once because there are domains that have more members than there are constants. E.g., the domain of real numbers has uncountably many members, whereas we have only countably many constants.[1] So we are not asking what would happen if everything

[1]This is also why we can't solve the problem by modifying the rules of the language so that every member of the domain has a name: that would rule out too many interesting collections of

had a name. Rather, we ask for each particular $\mathfrak{m}$ what would happen if we introduced a new constant for *it*.

So here is the more precise formulation for when a universally quantified sentence is true: $\forall \upsilon \varphi(\upsilon)$ is true in $\mathfrak{I}$ iff. each member $\mathfrak{m}$ of the domain is such that if κ is a new constant referring to $\mathfrak{m}$, $\varphi(\kappa)$ is true (alternatively, there is no member $\mathfrak{m}$ of the domain such that if κ is a new constant referring to $\mathfrak{m}$, $\varphi(\kappa)$ is false).

Similarly, $\exists \upsilon \varphi(\upsilon)$ is true in $\mathfrak{I}$ iff. there is some member $\mathfrak{m}$ such that if κ is a new constant referring to $\mathfrak{m}$, then $\varphi(\kappa)$ is true.

You might be puzzled by the fact these specifications of when quantified sentences are true do not mention the 'old' constants. Do they not matter? The point to notice is that if κ_1 and κ_2 are constants referring to the same thing in the domain, then $\varphi(\kappa_1)$ is true in $\mathfrak{I}$ iff. $\varphi(\kappa_2)$ is true in $\mathfrak{I}$.

We now have everything we need to figure out whether or not any sentence of $\mathcal{L}_Q$ is true in a given interpretation $\mathfrak{I}$. One thing worth noting is this: whether or not a sentence s is true in $\mathfrak{I}$ is solely a matter of the referents and extensions of the constants and predicates occurring in s. This will be of some use when discussing inferences in predicate logic.

This all may sound a bit abstract and scary. Let's discuss some examples to make things more concrete.

5.5.1 Examples

But first: we defined $\mathcal{L}_Q$ to have infinitely many constants and predicates. However, we are usually interested in only a small number of them, namely those that appear in the sentences whose truths we are interested in. Since the semantic of $\mathcal{L}_Q$ is such that the truth of a sentence in an interpretation only depends on the interpretation of the constants and predicates that occur in the sentence, we can safely ignore all the other constants and predicates. So in the following examples, only a small number of constants and predicates have their interpretations specified.

---

things from being the domain—like the real numbers.

Example 1

Let our language have 3 constants of interest: j, t, h. And let it have two one-place predicates of interest: F, R.

The following specifies our interpretation $\mathfrak{I}$:

Domain of Discourse: Jerry the mouse, Tom the cat, Hobbes the tiger.

Referents of Constants: j refers to Jerry, t refers to Tom, h refers to Hobbes.

Extensions or Predicates: The extension of F is {Tom, Hobbes}; the extension of R is {Jerry}

Let's consider some sentences.

- Ft is true in $\mathfrak{I}$ because the referent of t, Tom, is in the extension of F.
- Rt is not true in $\mathfrak{I}$ because the referent of t, Tom, is not in the extension of R.
- Rj is true in $\mathfrak{I}$ because the referent of j, Jerry, is in the extension of R.
- Fj is not true in $\mathfrak{I}$ because the referent of j, Jerry, is not in the extension of F.
- $\neg Fj$ is true in $\mathfrak{I}$ because Fj is not true in $\mathfrak{I}$.
- $Rj \supset Rt$ is false in $\mathfrak{I}$ because neither $\neg Rj$ nor Rt is true in $\mathfrak{I}$.
- $\forall x Fx$ is not true in $\mathfrak{I}$ because there is a member of the domain, Jerry, such that if we introduce c_{Jerry} as a constant referring to Jerry, Fc_{Jerry} is not true in the extended interpretation.
- $\forall x(Fx \vee Rx)$ is true in model because each member $\mathfrak{m}$ of the domain is such that if we introduce a new constant c_{new} to refer to it, $Fc_{new} \vee Rc_{new}$ is true in the extended interpretation.

Example 2

Same language as above but with the following interpretation $\mathfrak{I}$:

Domain of Discourse: Jar Jar Binks, Harry.

Referents of Constants: j refers to Jar Jar Binks, t refers to Jar Jar Binks, h refers to Harry.

Extensions of Predicates: The extension of F is {Jar Jar Binks, Harry}. The extension of R is empty.

Let's consider some sentences:

- Ft is true in $\mathfrak{I}$ because the referent of t, Jar Jar Binks, is in the extension of F.
- Rt is not true in $\mathfrak{I}$ because the referent of t, Jar Jar Binks, is not in the extension of R.
- Rj is not true in $\mathfrak{I}$ because the referent of j, Jar Jar Binks, is not in the extension of R.
- Fj is true in $\mathfrak{I}$ because the referent of j, Jar Jar Binks, is in the extension of F.
- $\neg Fj$ is not true in $\mathfrak{I}$ because Fj is true in $\mathfrak{I}$.
- $Rj \supset Rt$ is true in $\mathfrak{I}$ because $\neg Rj$ is true in $\mathfrak{I}$.
- $\forall x Fx$ is true in $\mathfrak{I}$ because each member $\mathfrak{m}$ of the domain is such that if we introduced a new constant $c_{\mathfrak{m}}$ to refer to it, $Fc_{\mathfrak{m}}$ is true in the extended interpretation (since all members of the domain are in the extension of F).
- $\forall x(Fx \vee Rx)$ is true in $\mathfrak{I}$ because each member $\mathfrak{m}$ of the domain is such that if we introduce a new constant $c_{\mathfrak{m}}$ to refer to it, $Fc_{\mathfrak{m}} \vee Rc_{\mathfrak{m}}$ is true in the extended interpretation.

Notice how changing the interpretation changes which sentences are true in the interpretation and which not.

Example 3

Same language as above but with the following interpretation:

Domain of Discourse: Jar Jar Binks, Harry, JFK.

Referents of Constants: j refers to Jar Jar Binks, t refers to Jar Jar Binks, h refers to Harry.

Extensions of Predicates: The extension of F is {Jar Jar Binks, Harry}. The extension of R is {JFK}.

for quantifier-free sentences, this interpretation behaves like the one in the previous example (check it). But:

- $\forall x Fx$ is not true in $\mathfrak{I}$ because introducing a new constant c_{JFK} to refer to JFK results in an extended interpretation in which Fc_{JFK} is not true because JFK is not in the extension of F .

5.6 Entailment, Logical Truth, Contradiction

Suppose $\forall x(Fx \wedge Gx)$ is true in $\mathfrak{I}$. In that case, $\forall xFx$ is also true in $\mathfrak{I}$. Suppose $\forall x(Fx \vee Gx)$ and $\forall x\neg Gx$ are both true in $\mathfrak{I}$. In that case, $\forall xFx$ is true in $\mathfrak{I}$. So the truth in $\mathfrak{I}$ of some sentences can guarantee truth of some sentence φ. We will write

$$\Gamma \vDash \varphi$$

to mean that every interpretation is such if all the sentences in Γ are true, then φ is also true. More colloquially, the truth of the sentences in Γ guarantees the truth of φ. We will read it as 'Γ entails φ', just as we did in sentential logic.

Some sentences are true in every possible interpretation. E.g.,$\forall x(Fx \vee \neg Fx)$ is true in every possible interpretation. A sentence that is true in every interpretation is called a *logical truth*. Any tautology of sentential logic is a logical truth, but not all logical truths are tautologies—just look at the one mentioned in this paragraph. We will write

$$\vDash \varphi$$

to mean that φ is a logical truth.

A logical falsehood is a sentence that is false in every possible interpretation.

We say that a set of sentences is *satisfiable* iff. if there is an interpretation $\mathfrak{I}$ such that all sentences in the set are true in $\mathfrak{I}$. Derivatively, we will say that an individual sentence is satisfiable iff. the set containing it as its unique member is satisfiable.

5.7 Reasoning With Quantifiers: Two Simple Rules

Let's go back to the argument in Example 5.1.2 (the one about Ed the whale). Our difficulty was that sentential logic could not account for the obvious cogency of that argument, and that seemed to be due to the limitations of $\mathcal{L}$. Let's see how $\mathcal{L}_Q$ can help.

Let e be a constant naming Ed, x a variable, and Wx and Mx mean 'x is a whale' and 'x is a mammal' respectively. We can formalize the argument as follows:

1.	Γ	$\vdash \forall x(Wx \supset Mx)$	premise
2.	Γ	$\vdash We \supset Me$	1,?
3.	Δ	$\vdash We$	premise
4.	Γ, Δ	$\vdash Me$	2,3,$\supset$E

The move from 1 to 2 is obviously cogent. If every x is such that if it is W, then it is M, it follows that any particular thing e is such that if it is W, it is M. We can turn this point into an inference rule:

Universal Quantifier Elimination ($\forall$E) From $\Lambda \vdash \forall \upsilon \varphi(\upsilon)$, infer $\Lambda \vdash \varphi(\kappa)$, for any constant κ. (All instances of υ must be replaced by κ.)

This is also often called Universal Instantiation because it formalizes the activity of giving an instance of a universal claim. We can now fill in the inference rule in the above argument:

1.	Γ	$\vdash \forall x(Wx \supset Mx)$	premise
2.	Γ	$\vdash We \supset Me$	1,$\forall$E
3.	Δ	$\vdash We$	premise
4.	Γ, Δ	$\vdash Me$	2,3,$\supset$E

Here is another obvious inference rule:

Existential Quantifier Introduction (∃I) Given a constant κ, infer from $\Lambda \vdash \varphi(\kappa)$ to $\Lambda \vdash \exists\upsilon\varphi(\upsilon)$. (At least one—but not necessarily all—instances of κ must be replaced by υ.)

This also is obvious: if κ is φ, then there is something that is φ, viz. κ. Here is an example illustrating the use of ∃I.

1.	Γ	$\vdash \forall x(Wx \supset Mx)$	..... premise
2.	Γ	$\vdash We \supset Me$	..... 1,∀E
3.	Δ	$\vdash We$	..... premise
4.	Γ, Δ	$\vdash Me$	..... 2,3,⊃E
5.	Γ, Δ	$\vdash \exists xMx$	..... 4,∃I

If Γ, Δ prove that Ed is a whale, then Γ, Δ prove that there is a whale.

Notice that ∃I allows the following inference:

1.	Γ	$\vdash Ldd$	..... premise
2.	Γ	$\vdash \exists xLxd$	..... 1,∃I

If Ldd means that Donald loves Donald, then it follows that there is someone (i.e., Donald) who loves Donald.

For the logical connectives of $\mathcal{L}$ we have a pair of rules for each of them. The same is true for the quantifiers. What we have so far is one half of the inference rules for quantifiers. Let's discuss the other two in the following sections.

5.8 Universal Quantifier Introduction

Take any constant c and a predicate F. We can easily prove $\vdash Fc \vee \neg Fc$. All we have to do is take our earlier proof of EM and replace p with Fc:

1.	$\neg(Fc \vee \neg Fc)$	$\vdash \neg(Fc \vee \neg Fc)$	A
2.	Fc	$\vdash Fc$	A
3.	Fc	$\vdash Fc \vee \neg Fc$	2,$\vee$I
4.	$Fc, \neg(Fc \vee \neg Fc)$	$\vdash Fc \vee \neg Fc$	3
5.	$\neg(Fc \vee \neg Fc), Fc$	$\vdash \neg(Fc \vee \neg Fc)$	1
6.	$\neg(Fc \vee \neg Fc)$	$\vdash \neg Fc$	4,5,$\neg$I
7.	$\neg(Fc \vee \neg Fc)$	$\vdash Fc \vee \neg Fc$	6,$\vee$I
8.		$\vdash \neg\neg(Fc \vee \neg Fc)$	1,7,$\neg$I
9.		$\vdash Fc \vee \neg Fc$	8,$\neg$E

Now, when you inspect the above argument, it is easy to see that we could run the same argument with any other constant. So we should be allowed to conclude $\vdash \forall x(Fx \vee \neg Fx)$.

What we just did is universalize from a particular case. That is usually a very bad idea (e.g., "Donald is narcissist, therefore everyone is a narcissist" is an obvious *non sequitur*[2]). So why does the above work? It is crucial that there is *nothing* special about c in the proof of $\vdash Fc \vee \neg Fc$. What does that mean? Different things differ from each other in various ways. But they also have things in common. To say that there was nothing special about c in the above proof is to say that nothing that makes c different from other things played a role in the proof. This is why we can be confident that the universalization to all x works.

So the thought is that if we can show that $\varphi(c)$ is true without depending on anything that distinguishes c from other things, then we can universalize to the claim that $\forall \upsilon \varphi(\upsilon)$. Let's put this as an inference rule:

[2] 'non sequitur' is Latin for 'it does not follow.'

Universal Quantifier Introduction ($\forall$I) Given a constant κ, from $\Lambda \vdash \varphi(\kappa)$ infer $\Lambda \vdash \forall \upsilon \varphi(\upsilon)$, provided κ does not appear in any of the sentences in Λ. (Must replace all instances of κ in $\varphi(\kappa)$ with υ.)

The proviso ensures that no claims about κ are needed to support $\varphi(\kappa)$ which means that nothing that might distinguish κ from other things plays a role in the support for $\varphi(\kappa)$.

We can now formalize the universalization of a tautology:

1. $\vdash Fc \vee \neg Fc$ ..... EM
2. $\vdash \forall x\big(Fx \vee \neg Fx\big)$ ..... 1,$\forall$I

So there are theorems involving quantifiers, too.

Notice that this introduces a sequent whose succedent is $Fc \vee \neg Fc$ without first declaring that c is a constant. If c is understood as a variable, this would not be a sentence and the sequent makes no sense. We will be charitable and interpret everything in such a way that any unbound term is understood as a constant with the exception that a term that is previously used as a variable somewhere in the derivation cannot be understood as a constant. And once a letter is used for a constant, you cannot use it as a name for a variable later. For example, take

1. $\Gamma \vdash \forall x Fx$ ..... premise
2. $\Gamma \vdash Fy$ ..... 1, $\forall$E

In this case, y on line 2 is understood as a constant. Because y does not appear as a variable previously. But the following is *not* ok:

1. $\Gamma \vdash \forall y Fy$ ..... premise
2. $\Gamma \vdash Fy$ ..... 1, $\forall$E

y is used as a variable on line 1, and that means that y must be understood as an unbound variable on line 2 which renders the sequent there senseless. Similarly, you cannot do

1.	Γ	$\vdash Fx$	premise
2.	Γ	$\vdash \exists x Fx$	1, $\exists$I

x on the first line must be understood as a constant, but then you cannot use it as a variable name on line 2.

Finally, we will adopt the convention that when a constant appears for the first time in a derivation, it is a *new* constant that is being added to $\mathcal{L}_Q$ unless it is explicitly specified otherwise in the context. Remember our discussion of the semantics of quantified sentences: it is possible to add new constants without affecting what is true without that new constant. All sentences that appear prior to the introduction of the constant should be understood as sentences in the language without the new constant. This makes life a lot easier when it comes to applying $\forall$I.

Here is an example: all whales are mammals, and all mammals are warm-blooded; it follows that all whales are warm-blooded. We can formalize this using the same interpretation of the predicate letters as as in the previous section:

1.	Γ	$\vdash \forall x(Wx \supset Mx)$	premise
2.	Δ	$\vdash \forall x(Mx \supset Bx)$	premise
3.	Wn	$\vdash Wn$	A
4.	Γ	$\vdash Wn \supset Mn$	1,$\forall$E
5.	Δ	$\vdash Mn \supset Bn$	2,$\forall$E
6.	Γ, Wn	$\vdash Mn$	3,4,$\supset$E
7.	Γ, Δ, Wn	$\vdash Bn$	5,6,$\supset$E
8.	Γ, Δ	$\vdash Wn \supset Bn$	7,$\supset$I
9.	Γ, Δ	$\vdash \forall x(Wx \supset Bx)$	8,$\forall$I

Here is an informal version of what this does: all whales are mammals, and all mammals are warm-blooded. Pick anything that's a whale. Let's call it Nina. Since Nina is a whale, Nina is a mammal, and therefore warm-blooded. Nothing depended on our choice of Nina. So all whales are warm-blooded.

In the derivation, we introduce a new constant n on line 3 with the assumption $Wn \vdash Wn$. We can tell it's a constant since if it were a variable the sequent would not make sense, and n does not occur in the previous lines as a variable. And it's a *new* constant because n appears there for the first time. Because n is a new constant, it cannot appear in Γ, Δ: those must be sets of sentences of the original language because they are introduced into the derivation before the new constant is introduced. So we know the proviso for $\forall$I is met for the inference to line 9.

There is one more rule of inference. It is, you guessed it, the Existential Quantifier Elimination rule. We will discuss that in the next section.

5.9 Existential Quantifier Elimination

Here is something obvious: if all whales are mammals, and there are whales, it follows that there are mammals. This means that we should be able to infer from $\Gamma \vdash \forall x(Wx \supset Mx)$ and $\Delta \vdash \exists x Wx$ to $\Gamma, \Delta \vdash \exists x Mx$.

The inference rules we have so far do not enable us to make the inference, though. So we need a new inference rule. What should that look like? Here is a thought. Consider the following argument:

> **Example 5.9.1.** All whales are mammals, and there are whales. Let's suppose Cindy is one of the whales. On this supposition, Cindy is a mammal because of the first premise. So, on the supposition that Cindy is a whale, there are mammals. Now, we don't know that Cindy is a whale. But there are whales and that means that what we did is like picking a whale and naming it Cindy. So whether or not the real Cindy is a whale, there are mammals.

We will call the reasoning pattern exhibited here Existential Quantifier Elimination. Given a general existential claim, we pretend that the claim is true of a particular thing, κ, show that something follows on that pretense, and conclude that it follows even without the pretense.

We do need to be careful. Consider:

> **Example 5.9.2.** There are scientist. So let's suppose Donald is a scientist. Donald believes that injecting bleach can cure infectious diseases. So on the supposition that Donald is a scientist, there are scientists who believe that injecting bleach can cure infectious diseases. Therefore, there are scientists who believe that injecting bleach can cure infectious diseases.

This is a terrible piece of reasoning. How does it differ from the Cindy case? The problem is that believing that injecting bleach can cure infectious diseases is a special fact about Donald. Had you picked somebody else, say Deborah, you wouldn't have been able to say that Deborah believes that injecting bleach can cure infectious diseases. In the Cindy case, the reasoning does not depend on anything special about Cindy. That's crucial.

Here is a rule of inference that captures the Cindy case while ruling out the Donald case:

Existential Quantifier Elimination (∃E) From $\Lambda_1 \vdash \exists \upsilon \varphi(\upsilon)$ and $\Lambda_2, \varphi(\kappa) \vdash \psi$, infer $\Lambda_1, \Lambda_2 \vdash \psi$, provided κ does not appear in any of Λ_1, Λ_2, and ψ.

A note on the proviso that κ must not appear in any of Λ_1, Λ_2, and ψ. Requiring that κ not appear in Λ_1 or Λ_2 ensures we do not illicitly depend on some special features of κ. By requiring that κ not appear in ψ, we prevent concluding something about κ even though Λ_1 and Λ_2 say nothing about κ.

We can now formalize the argument in the above Example involving Cindy:

1. $\Gamma \vdash \forall x(Wx \supset Mx)$ ..... premise
2. $\Delta \vdash \exists y Wy$ ..... premise
3. $Wc \vdash Wc$ ..... A
4. $\Gamma \vdash Wc \supset Mc$ ..... 1,∀E
5. $\Gamma, Wc \vdash Mc$ ..... 3,4,⊃E
6. $\Gamma, Wc \vdash \exists z Mz$ ..... 5,∃I
7. $\Gamma, \Delta \vdash \exists z Mz$ ..... 2,6,∃E

Here is another example of ∃E in action. Suppose there is something that is F. It follows that not everything is not-F. E.g., if there are reasonable people, not everyone is unreasonable. That means we should be able to infer from $\Gamma \vdash \exists x Fx$ to $\Gamma \vdash \neg\forall x \neg Fx$. Here is one way: Assume that something is F, and also assume that everything is not-F. Let Alec be F. But given the second assumption, Alec is not-F. But then Alec is both F and not-F. That's a contradiction. Something has to give. Since we are going to keep the first assumption, we must give up the second assumption. Thus, not everything is not-F.

We can formalize this:

1. $\Gamma \vdash \exists x Fx$ ..... premise
2. $\forall x \neg Fx \vdash \forall x \neg Fx$ ..... A
3. $Fa \vdash Fa$ ..... A
4. $\forall x \neg Fx \vdash \neg Fa$ ..... 2,∀E

5. $Fa, \forall x \neg Fx \vdash \neg Fa$ 4

6. $Fa, \forall x \neg Fx \vdash Fa$ 3

7. $Fa \vdash \neg \forall x \neg Fx$ . . . . . 5,6,¬I

8. $\Gamma \vdash \neg \forall x \neg Fx$ . . . . . 1,7,∃E

5.10 Proof System for Predicate Logic

Here is our proof system for predicate logic in one place. We now use Greek lower case letters for sentence variables. You may:

Assumption Introduction (A) Infer $\varphi \vdash \varphi$.

Conjunction Introduction ($\wedge$I) From $\Lambda_1 \vdash \varphi_1$ and $\Lambda_2 \vdash \varphi_2$, infer $\Lambda_1, \Lambda_2 \vdash \varphi_1 \wedge \varphi_2$.

Conjunction Elimination ($\wedge$E) From $\Lambda \vdash \varphi_1 \wedge \varphi_2$, infer $\Lambda \vdash \varphi_1$ as well as $\Lambda \vdash \varphi_2$.

Disjunction Introduction ($\vee$I) From $\Lambda \vdash \varphi_1$, infer $\Lambda \vdash \varphi_1 \vee \varphi_2$ as well as $\Lambda \vdash \varphi_2 \vee \varphi_1$, for any φ_2.

Disjunction Elimination ($\vee$E) From $\Lambda_1 \vdash \varphi_1 \vee \varphi_2$ and $\varphi_1, \Lambda_2 \vdash \varphi_3$ and $\varphi_2, \Lambda_3 \vdash \varphi_3$, infer $\Lambda_1, \Lambda_2, \Lambda_3 \vdash \varphi_3$.

Negation Introduction ($\neg$I) From $\Lambda_1, \varphi_1 \vdash \varphi_2$ and $\Lambda_2, \varphi_1 \vdash \neg\varphi_2$, infer $\Lambda_1, \Lambda_2 \vdash \neg\varphi_1$.

Negation Elimination ($\neg$E) From $\Lambda \vdash \neg\neg\varphi$, infer $\Lambda \vdash \varphi$.

Conditional Introduction ($\supset$I) From $\Lambda, \varphi_1 \vdash \varphi_2$, infer $\Lambda \vdash \varphi_1 \supset \varphi_2$.

Conditional Elimination ($\supset$E) From $\Lambda_1 \vdash \varphi_1 \supset \varphi_2$ and $\Lambda_2 \vdash \varphi_1$, infer $\Lambda_1, \Lambda_2 \vdash \varphi_2$.

Universal Quantifier Introduction ($\forall$I) Given a constant κ, from $\Lambda \vdash \varphi(\kappa)$ infer $\Lambda \vdash \forall\upsilon\varphi(\upsilon)$, provided κ does not appear in any of the sentences in Λ. (Must replace all instances of κ in $\varphi(\kappa)$ with υ.)

Universal Quantifier Elimination ($\forall$E) From $\Lambda \vdash \forall\upsilon\varphi(\upsilon)$, infer $\Lambda \vdash \varphi(\kappa)$, for any constant κ. (All instances of υ must be replaced by κ.)

Existential Quantifier Elimination ($\exists$E) From $\Lambda_1 \vdash \exists\upsilon\varphi(\upsilon)$ and $\Lambda_2, \varphi(\kappa) \vdash \psi$, infer $\Lambda_1, \Lambda_2 \vdash \psi$, provided κ does not appear in any of Λ_1, Λ_2, and ψ.

Existential Quantifier Introduction ($\exists$I) Given a constant κ, infer from $\Lambda \vdash \varphi(\kappa)$ to $\Lambda \vdash \exists\upsilon\varphi(\upsilon)$. (At least one—but not necessarily all—instances of κ must be replaced by υ.)

Also, you may rewrite the datum side of sequents in the following ways:

a. You may reorder items within the datum as you see fit.

b. You may delete duplicate items within the datum.

c. You may add arbitrary items to the datum of a sequent.

Finally, you may introduce new constants into a derivation using Assumption Introduction or Universal Quantifier Elimination. The new constant must not appear on any previous line nor be declared a constant in the surrounding context.

5.11 Theorems

As we have already seen, predicate logic has theorems, too. A proof proves a theorem if it proves a sequent with an empty datum by using only the rules of our proof system. It is exactly like what it was in the case of sentential logic. And we can prove theorems in just the same sort of ways as before. E.g.,

1. $\exists x Fx \vdash \exists x Fx$ ..... A
2. $\forall x \neg Fx \vdash \forall x \neg Fx$ ..... A
3. $Fa \vdash Fa$ ..... A
4. $Fa, \forall x \neg Fx \vdash Fa$ 3
5. $\forall x \neg Fx \vdash \neg Fa$ ..... 2,$\forall$E
6. $Fa \vdash \neg \forall x \neg Fx$ ..... 4,5,$\neg$I
7. $\exists x Fx \vdash \neg \forall x \neg Fx$ ..... 1,6,$\exists$E
8. $\vdash \exists x Fx \supset \neg \forall x \neg Fx$ ..... 7,$\supset$I

5.11.1 More Theorems

Here are some useful theorems:

Quantifier Exchange (QE)
$\exists x Fx \supset \neg \forall x \neg Fx$

Quantifier Exchange (QE)
$\neg \forall x \neg Fx \supset \exists x Fx$

Quantifier Exchange (QE)
$\forall x Fx \supset \neg \exists x \neg Fx$

Quantifier Exchange (QE)
$\neg \exists x \neg Fx \supset \forall x Fx$

Quantifier Exchange (QE)
$\exists x \neg Fx \supset \neg \forall x Fx$

Quantifier Exchange (QE)
$\neg\forall xFx \supset \exists x\neg Fx$

Quantifier Exchange (QE)
$\forall x\neg Fx \supset \neg\exists xFx$

Quantifier Exchange (QE)
$\neg\exists xFx \supset \forall x\neg Fx$

Confinement (CF)
$\big(\forall xFx \wedge \forall xGx\big) \supset \forall x\big(Fx \wedge Gx\big)$

Confinement (CF)
$\forall x\big(Fx \wedge Gx\big) \supset \big(\forall xFx \wedge \forall xGx\big)$

Confinement (CF)
$\big(\forall xFx \vee \forall xGx\big) \supset \forall x\big(Fx \vee Gx\big)$

Confinement (CF)
$\big(\exists xFx \wedge \exists xGx\big) \supset \exists x\exists y\big(Fx \wedge Gy\big)$

Confinement (CF)
$\exists x\exists y\big(Fx \wedge Gy\big) \supset \big(\exists xFx \wedge \exists xGx\big)$

Confinement (CF)
$\big(\exists xFx \vee \exists xGx\big) \supset \exists x\big(Fx \vee Gx\big)$

Confinement (CF)
$\exists x\big(Fx \vee Gx\big) \supset \big(\exists xFx \vee \exists xGx\big)$

5.12 Generalizing the Theorems

Take a look at the theorem we proved in the last section. The proof takes a particular predicate F but it is obvious that you could replace it with any other predicate and construct a parallel proof. So just like the proofs we saw of theorems of sentential logic, the proof of that Theorem (and the other theorems) is a proof template. You can replace any 1-place predicate with any other 1-place predicate.

You will notice that the sentences involved in that proof of the last section are very simple. For instance, $\exists x F x$ has an existential quantifier attached to the simplest formula possible. But quantifiers can also be attached to complex formulas. We can easily generalize the proof in the previous section to cover cases in which the existential quantifier is attached to a complex formula by replacing Fx with $\varphi(x)$ and Fa with $\varphi(a)$ (a is a constant). $\varphi(x)$ is a formula in which x is free and $\varphi(a)$ is the formula obtained by substituting a into x:

1. $\exists x\varphi(x) \vdash \exists x\varphi(x)$ ..... A
2. $\forall x\neg\varphi(x) \vdash \forall x\neg\varphi(x)$ ..... A
3. $\varphi(a) \vdash \varphi(a)$ ..... A
4. $\varphi(a), \forall x\varphi(x) \vdash \varphi(a)$ 3
5. $\forall x\neg\varphi(x) \vdash \neg\varphi(a)$ ..... 2,$\forall$E
6. $\varphi(a) \vdash \neg\forall x\neg\varphi(x)$ ..... 4,5,$\neg$I
7. $\exists x\varphi(x) \vdash \neg\forall x\neg\varphi(x)$ ..... 1,6,$\exists$E
8. $\vdash \exists x\varphi(x) \supset \neg\forall x\neg\varphi(x)$ ..... 7,$\supset$I

The proofs for the other theorems can be generalized in the same fashion. For the rest of the course, you may assume that we have proven the theorems in this generalized form.

5.13 Soundness of the System of Predicate Logic

Given the explanation of the inference rules, it is plausible that our proof system is sound. We will use the same terminology as in Section 4.7: $\Gamma \vdash \varphi$ is correct iff. $\Gamma \vDash \varphi$. What we want to show is that if a derivation consists of only correct sequents, then extending the proof by one more line via one of our four new rules will result in another correct sequent.

5.13.1 Existential Quantifier Introduction

This is easy to see given the semantics of the existential quantifier. Any interpretation in which $\varphi(c)$ is true is also such that $\exists x \varphi(x)$ is true.

5.13.2 Universal Quantifier Elimination

This, too is easy to see given the semantics of the universal quantifier.

5.13.3 Universal Quantifier Introduction

Suppose $\Gamma \vDash \varphi(c)$ where c does not appear in any of the sentences of Γ. Let $\mathfrak{I}$ be an interpretation in which all the sentences in Γ are true. We are supposing that $\varphi(c)$ is also true in $\mathfrak{I}$. Consider an interpretation $\mathfrak{I}'$ which is just like $\mathfrak{I}$ except that it assigns a different referent to c. All the sentences in Γ must still be true in $\mathfrak{I}'$ because c does not appear in any sentence in Γ so that the referent of c makes no difference to the truth of those sentences. So $\varphi(c)$ must also be true in $\mathfrak{I}'$ because $\Gamma \vDash \varphi(c)$. But this means that any interpretation that makes all sentences in Γ true also make $\varphi(c)$ true independently of the referent of c. Thus, if $\Gamma \vDash \varphi(c)$, then $\Gamma \vDash \forall x \varphi(x)$ provided c does not appear in any sentence in Γ. This shows that $\forall$I is a valid rule of inference.

5.13.4 Existential Quantifier Elimination

Suppose we can prove $\Delta, \varphi(c) \vdash \psi$ where c does not appear in Δ or ψ and that the sequent is correct. That is, $\Delta, \varphi(c) \vDash \psi$. The point to notice is that this is a claim about truth. What it says is that if $\varphi(c)$ is true, then $\varphi(\kappa)$ is true for any constant κ that refers to the same thing as c. I.e., if an interpretation is such

that the sentences in Δ are true and there is something that is φ (if κ refers to it, then $\varphi(\kappa)$ is true), then ψ is also true in that interpretation.

With this in mind, suppose we also have $\Gamma \vDash \exists x \varphi(x)$, and consider an interpretation $\mathfrak{I}$ that makes all sentences in Γ and Δ true. Given the semantics of the existential quantifier, $\mathfrak{I}$ is such that there is something in the domain that is φ. Thus, $\mathfrak{I}$ also makes ψ true: $\Gamma, \Delta \vDash \psi$. This shows that $\exists$E is a valid rule of inference.

5.13.5 Completeness

What about completeness? Can our system prove all truths of logic?

As stated, our system is not complete. Here is a truth of logic that cannot be proved in our system:

$$Fc \supset \forall x Fc$$

Notice that the universal quantifier does not bind anything in its scope. We have been discouraging such formulas but it is allowed. Given the semantics of the universal quantifier, $\forall x Fc$ is equivalent to Fc so the above is indeed a truth of logic.

We can modify Universal Quantifier Introduction a bit to allow inferring from $\Lambda \vdash \varphi$ to $\Lambda \vdash \forall \upsilon \varphi$ (notice that φ is unmodified). This modified system is complete but we will not be able to prove it here. I have to ask you to take my word for it.

If you are interested, Kurt Gödel (1929) was the first person to prove the completeness of predicate logic. The proof was his doctoral dissertation. Nowadays the preferred proof of completeness is one that was given by Leon Henkin (1949).

5.14 Counting and Identity

The sentence $\exists x Fx$ tells us that there is at least one thing that is F. But it does not tell us how many. Maybe there is only one, but maybe there are gazillions. $\exists x Fx$ does not distinguish between those cases. How could we say there is exactly one thing which is F?

Let's start with an easier question. How could we say that there are at least two things that are F? To say that there are two things that are F is to say that there are two distinct things that are F. We might try:

1. $\exists x Fx \wedge \exists y Fy$

There is an x which is F, and there is a y which is F. But this does not work. $\exists x Fx$ is true just in case there is at least one thing that is F. And the same goes for $\exists y Fy$. They are two ways of saying the same thing. So even if there is only one thing that is F, 1 would be true. We need to ensure that x and y are different things. Let us write $x \neq y$ to mean that x is not identical to y. And we will write $x = y$ to mean that x is identical to y. That is, we will use the identity sign the way it is normally used. We could then try:

2. $\big(\exists x Fx \wedge \exists y Fy\big) \wedge x \neq y$

But this still does not work because the formula $x \neq y$ is outside the scope of either quantifier. What we want to say is something like: we can find an x which is F, and find a y which is F, and when we compare x and y we find that they are two distinct things. 2 doesn't say that. To be able to compare the two things we find, we have to get them within the scope of the same quantifier. The following will do:

3. $\exists x\big[Fx \wedge \exists y\big(Fy \wedge x \neq y\big)\big]$

This says that there is an x which is F and also a y which is F but is different from x. That means there are at least two things that are F. It's the existential claim starting with $\exists y$ inside the scope of the $\exists x$ that guarantees the existence of a second thing that is F.

As an example, suppose our domain contains only mom, dad, grandma, and Fido. Fido is a dog, the others are humans; dad is male, the other humans in

the domain are women. As you can see, there are at least two women in our domain: mom and grandma. And if we let Fx mean 'x is a woman' we can see that the above sentence is true given our domain of discourse. There is someone, e.g. mom, who is a woman and there is someone else, e.g., grandma, who is a woman but is not mom.

Let's get back to the original question, How do we say that there is exactly one thing which is F? Consider Fido. Fido is the only dog in our domain so there is exactly one dog in the domain. Since we now know how we could say that there are at least two dogs in the domain, we can also say that there is exactly one dog: there are dogs, but it is not true that there are at least two dogs. And that means there is a thing which is a dog but there is no second thing that is a dog. More generally, there is exactly one F is to say that there is at least one things that is F but there is no second thing that is F. We can say this using the following sentence:

4. $\exists x\big[Fx \wedge \neg\exists y\big(Fy \wedge x{\neq}y\big)\big]$

Notice the negation sign in front of the $\exists y$. That rules out that there is a second thing that is F. Alternatively, we could also say:

5. $\exists x\big[Fx \wedge \forall y(Fy \supset x{=}y)\big]$

Continuing with our example domain of discourse, we can see that there are exactly two women: mom and grandma. How could we say that in our formal language? That is not hard. We start with the sentence that says there are at least two and add something that rules out the existence of a third. Like this:

6. $\exists x\Big(Fx \wedge \exists y\big\{\big(Fy \wedge x{\neq}y\big) \wedge \neg\exists z\big[\big(Fz \wedge x{\neq}z\big) \wedge y{\neq}z\big]\big\}\Big)$

There is an x which is F, another thing y which is also F, but there is no further thing distinct from both x and y that is also F.

We can say that there are at least three by removing the negation sign in the sentence above:

7. $\exists x\Big(Fx \wedge \exists y\big\{\big(Fy \wedge x{\neq}y\big) \wedge \exists z\big[\big(Fz \wedge x{\neq}z\big) \wedge y{\neq}z\big]\big\}\Big)$

We can then say that there are exactly three by adding a clause that rules out the existence of any further thing that is F. You get the drift. We could

continue that forever. So for every natural number n, we can say that there are exactly n things that are F as well as that there are at least n things that are F. Of course, we will run out of letters of the Roman alphabet quickly, so will have to resort to subscripts. The genius of the Arabic numerals guarantees we will never run out of them.

5.15 Axioms

When I introduced the identity sign in the previous section, I cheated a bit by depending on your previous understanding of the = symbol. Let's a look at it more closely.

A formula like $x{=}y$ expresses a relation between x and y. So the identity sign is just a 2-place predicate. We could have kept our notation more consistent by writing $=xy$ but that looks very unfamiliar, so we (like everybody else) will use the more familiar style. Since = is just a 2-place predicate, if we treat it like any other 2-place predicate, there is no reason why it should mean identity. In declaring that the symbol is to be used in familiar ways, I imposed a certain interpretation of it. In other words, I restricted the permissible interpretations of our language to those that take = to mean identity. What are the features of a interpretation that take = to mean identity?

We can specify the requirements in terms of the sentences that must come out to be true. Any interpretation that takes = to mean identity must make all sentences that fit the following templates true:

1. $x{=}x$
2. $\big(x{=}y \wedge \varphi(x)\big) \supset \varphi(y)$

That is, everything is identical with itself, and if x and y are identical, whatever is true of x is true of y (the latter is known as the indiscernibility of identicals, or Leibniz's Law).

Sentences that are declared to be true are called *axioms*. So the above templates give us an infinite number of axioms. We want to make sure that our system can make use of these axioms. We can accomplish that by adding the following inference rule to our system:

Axiom If α is an axiom, infer $\vdash \alpha$.

Notice that the sequents that this rule allows us to infer have empty datums. However, the succedents of these sequents are not truths of logic. For instance, we can infer $\vdash \kappa{=}\kappa$ but $\kappa{=}\kappa$ is true only under a particular interpretation of the = sign.

The characterization of an axiom as a sentence declared to be true without proof may make it sound like an assumption. But the role of an axiom is im-

portantly different from an assumption. In our proof system, an assumption is introduced as $s \vdash s$ which is a correct sequent under any and all interpretations. So in introducing an assumption, we say nothing about which interpretation we want to use. An axiom is different. Making an axiom true—or, equivalently, ensuring that Axiom does not lead us astray by enabling us to infer a falsehood—requires that we use some interpretations but not others as can be seen for the need to interpret = as identity given the axioms governing =. In the case identity, it is trivial to ensure = means identity no matter what the domain of discourse is: the extension of = is simply the set of all ordered pairs $< \mathfrak{m}, \mathfrak{m} >$ where $\mathfrak{m}$ is a member of the domain. Because of this, the axioms governing = are typically added to the proof system to yield what is known as predicate logic *with identity*.

But we can also add axioms that impose much more serious constraints in the permissible interpretations. Here is an intuitive example: we might declare that the following as axioms:

1. $\forall x \exists y \big(Tyx \wedge x \neq y \big)$
2. $\forall x \forall y \forall z \big[\big(Txy \wedge Tyz \big) \supset Txz \big]$
3. $\forall x \forall y \big(Txy \supset \neg Tyx \big)$

Tthese three combined require an infinite domain of discourse—e.g., if Tab means that a is taller than b, then those axioms require the existence of ever taller items in the domain.

Axioms like the ones just discussed play an important role in mathematics. Such axioms are used to define the subject matter by imposing requirements that can only be met by some domains but not others. For instance, Euclidean geometry (the kind you learned pre-college) is built on five axioms that together determine the kinds of geometric figures that Euclidean geometry deals with. Given those axioms, Euclidean geometry is not talking about figures drawn on the surface of a sphere. If you change an axiom you get a different subject matter. In particular, changing the so-called parallel postulate results in various interesting non-Euclidean geometries. An axiom therefore imposes restrictions on the interpretations in ways that assumptions cannot.

When formalizing arguments as derivations, I resisted treating ordinary premises as assumptions. Here is one way of thinking about premises. A premise takes the form $\Gamma \vdash p$ and, if Γ is non-empty, that's the same as $\vdash \widehat{\Gamma} \supset p$

where $\widehat{\Gamma}$ is the conjunction of all the sentences of Γ. So we can think of the introduction of a premise as the introduction of an axiom. Unlike an assumption, a premise makes a substantive claim about the world and we are to choose an interpretation for our language that makes that claim true.

Bibliography

Allen, Colin, and Michael Hand. 2001. *Logic Primer*. second edition ed., Cambridge, Massachusetts: MIT Press.

Cantor, Georg. 1891. Ueber eine elementare Frage der Mannigfaltigkeitslehre. *Jahresbericht Der Deutschen Mathematiker-Vereinigung* 72–78.

———. 1895. Beiträge zur Begründung der transfiniten Mengenlehre. *Mathematische Annalen* 46 (4): 481–512.

Gentzen, Gerhard. 1935a. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift* 39 (2).

———. 1935b. Untersuchungen über das logische Schließen II. *Mathematische Zeitschrift* 39 (3).

———. 1936. Die Wiederspruchsfreiheit der reinen Zahlentheorie. *Mathematische Annalen* 112:493–565.

Gödel, Kurt. 1929. *Über die Vollständigkeit des Logikkalküls*. Dissertation. Universität Wien.

Henkin, Leon. 1949. The completeness of the first-order functional calculus. *The Journal of Symbolic Logic* 14 (3): 159–166.

Hilbert, D. 1926. Über das Unendliche. *Mathematische Annalen* 95:161–190.

Hunter, Geoffrey. 1996. *Metalogic: An Introduction to the Metatheory of Standard First Order Logic*. University of California Press.

Kalmár, László. 1935. Über die Axiomatisierbarkeit des Aussagenkalküls. *Acta Scientiarum Mathematicarum* 7:222–243.

Lemmon, E. J. 1978. *Beginning Logic*. Indianapolis: Hackett Pub. Co.