forallχ (UBC edition)

An Introduction to Formal Logic
this copy compiled July 4, 2020

Jonathan Jenkins Ichikawa
University of British Columbia
P.D. Magnus
University at Albany, State University of New York

Jonathan Ichikawa, who modified the Magnus edition to create this version, would, first and foremost, like to thank P.D. Magnus. He is also grateful to Greg Restall, and his book *Logic*, which is the first text he taught from and which is the model for the discussion of trees. He also thanks the many UBC students who test-drove drafts of the book. He's also grateful for helpful conversations with Roberta Ballarin, Roy Cook, Josh Dever, Dave Gilbert, Thony Gillies, and Jon Shaheen.

The original author, P.D. Magnus, would like to thank the people who made this project possible. Notable among these are Cristyn Magnus, who read many early drafts; Aaron Schiller, who was an early adopter and provided considerable, helpful feedback; and Bin Kang, Craig Erb, Nathan Carter, Wes McMichael, Selva Samuel, Dave Krueger, Brandon Lee, Toan Tran, and the students of Introduction to Logic, who detected various errors in previous versions of the book.

Preface to the UBC Edition

This preface outlines my approach to teaching logic, and explains the way this version of *forall x* differs from Magnus's original. The preface is intended more for instructors than for students.

I have been teaching logic at the University of British Columbia since 2011; starting in 2017, I decided to prepare this textbook, based on and incorporating much of P. D. Magnus's *forall x*, which has been freely available for use and modification since 2005. Preparing this text had two main advantages for me: it allowed me to tailor the text precisely to my teaching preferences and emphasis, and, because it is available for free, it is saving money for students. (I encourage instructors to take this latter consideration pretty seriously. If you have a hundred students a year, requiring them each to buy a $50 textbook takes $5,000 out of students' pockets each year. If you teach with this or another free book instead, you'll save your students $50,000 over ten years. It can be sort of annoying to switch textbooks if you're used to something already. But is staying the course worth $50,000 of your students' money?)

This text was designed for a one-semester, thirteen-week course with no prerequisites, introducing formal logic. At UBC, the course has quite a mix of students with diverse academic backgrounds. For many it is their first philosophy course. As I teach Introduction to Formal Logic, the course has three central aims: (1) to help students think more clearly about arguments and argumentative structure, in a way applicable to informal arguments in philosophy and elsewhere; (2) to provide some familiarity and comfort with formal proof systems, including practice setting out formal proofs with each step justified by a syntactically-defined rule; and (3) to provide the conceptual groundwork for metatheoretical proofs, introducing the ideas of rigorous informal proofs about formal systems, preparing students for possible future courses focusing on metalogic and computability. I try to give those three elements roughly equal focus in my course, and in this book.

The book introduces two different kinds of formal proof systems — analytic tableaux ('trees') and Fitch-style natural deduction. Unlike many logic texts,

it puts its greater emphasis on trees. There are two reasons I have found this to be useful. One is that the algorithmic nature of tree proofs means that one can be assured to achieve successful proofs on the basis of patience and careful diligence, as opposed to requiring a difficult-to-quantify (and difficult-to-teach) 'flash of insight'. The other is that the soundness and completeness theorems for tree methods are simpler and more intuitive than they are for natural deduction systems, and I find it valuable to expose students to proofs of significant metatheoretical results early in their logical studies. (I prove soundness and completeness for a sentential logic tree system in the fifth week of the semester.) As presented here, the soundness and completeness proofs emphasize contrasting the systems students learn with hypothetical alternative systems that modify the rules in various ways. A rule like this would undermine the soundness of the system, but not its completeness. If we changed the rules in this way, it would still be both sound and complete. Etc. This helps give intuitive substance to these theorems.

I also include a Fitch-style natural deduction system, both for sentential and quantified logic, both because its premise-conclusion form is particularly helpful for thinking about informal arguments, and because it is important to recognize and follow proofs laid out in that kind of format, for example in more advanced philosophical material. While students do learn to do Fitch-style proofs, I emphasize less of that puzzle-solving kind of skill here than in many textbooks.

The book begins with a systematic engagement with sentential logic in conventional ways: translations, sentential connectives, models, truth tables, and both proof systems, including soundness and completeness for the tree system. Students are thereby able to become familiar with all the central metalogical ideas, incorporating relatively simple logical symbolism, before introducing predicates, quantifiers, and identity. Once we enrich the language, we go through those previous ideas again, using our more complex vocabulary.

The first book I used for teaching was Greg Restall's *Logic* (McGill–Queen's University Press, 2006), which I used for several years. My approach to teaching logic is heavily informed by that book; its influence in this text is particularly clear in the discussion of trees. (The natural deduction system I use is rather different from Restall's.)

In preparing this text, I began with Magnus's original and edited freely. There are sections where Magnus's prose has been retained entirely, and many of the exercises I have taken unchanged from the original. But I have also restructured many things and added quite a bit of new material. Unlike my version, which focuses on sentential logic before introducing predicates and quantification, Magnus's version integrated the discussion of sentential and quantificational systems, e.g. covering translation for both before discussing models and proofs for either. The original also did not include trees or soundness and completeness proofs. The two chapters on trees (5 and 10) and soundness and completeness (6

and 11) were written from scratch; my chapter on identity (12) is also original. The other material in this edition incorporates Magnus's original material, some parts more heavily edited than others. I have slightly modified Magnus's natural deduction rules.

In December 2018 I released the 1.0 version of this text. I'd had 'beta' versions available online for a couple years while teaching with them, but the present version represents a relatively stable version. A few small corrections since then have been implemented in this 1.2 version, as of October 2019.

Many thanks, first and foremost, to P.D. Magnus for providing this wonderful resource under a Creative Commons license, which made it freely available and gave me the right to modify and distribute it under the same licensing agreement. I hope other instructors will also feel free to either teach directly from this version, or to modify it to develop their own. The typesetting for trees is via Clea F. Rees's prooftrees package; thanks to her for making it available.

I'm grateful to the students in my 2017–20 PHIL 220 courses at UBC, who had an in-progress version of this book as their course textbook. They patiently and helpfully found and pointed out mistakes as I wrote them (incentivized, perhaps, by an offer of extra credit); this version has many fewer errors than it otherwise would have had. Thanks also to Cavell Chan and Joey Deeth, who did careful proofreading, and generated many solutions to exercises for the answer key, and to Laura Greenstreet for LaTeX and other technical help. These three assistants were supported by a UBC Library Open Access Grant in 2018–19.

I am maintaining a list of known issues and errors for this book, to be corrected in future editions, under 'issues' at `https://github.com/jonathanichikawa/for-all-x`. If you see any mistakes, please feel free to add them there directly, or to email me with them. The most recent version of the book is also always available for download there too.

<div align="right">

Jonathan Ichikawa
University of British Columbia
October 2019
ichikawa@gmail.com

</div>

Contents

Chapter 1

What is logic?

Logic is the business of evaluating arguments, sorting good ones from bad ones. In everyday language, we sometimes use the word 'argument' to refer to belligerent shouting matches. If you and a friend have an argument in this sense, things are not going well between the two of you. This is not the kind of 'argument' that will concern us. Arguments in the logical sense aren't events that happen between people; a logical argument is structured to give someone a reason to believe some conclusion. Here are two examples of such arguments:

(1) It is raining heavily.
(2) When it rains, everyone outside without an umbrella gets wet.
∴ You should take an umbrella.

(1) It is either raining or snowing.
(2) If it is colder than -10 degrees, it is not raining.
(3) It is -18 degrees.
∴ It is snowing.

The three dots on the last line of each argument mean 'Therefore' and they indicate that the final sentence is the *conclusion* of the argument. The other sentences are *premises* of the argument. If you believe the premises, then the argument provides you with a reason to believe the conclusion.

This chapter discusses some basic logical notions that apply to arguments in a natural language like English. It is important to begin with a clear understanding of what arguments are and of what it means for an argument to be valid. Later we will translate arguments from English into a formal language. We want formal validity, as defined in the formal language, to have at least some of the important features of natural-language validity.

1.1 Arguments

A crucial part of analyzing an argument is identifying its conclusion. Every argument has a conclusion — the conclusion is the claim the argument is trying to establish. Premises are starting-points, used to lend support to the conclusion. Often, the conclusion will be signified by words like 'so' or 'therefore'. Premises might be marked by words like 'because'. These words can give a clue as to just what the argument is supposed to be.

premise indicators: since, because, given that

conclusion indicators: therefore, hence, thus, then, so

In a natural language like English, *sometimes*, arguments start with their premises and end with their conclusions, but not always. For some purposes in this course, we will be working with *idealizations* of natural language, where we work as if some generally applicable rules of thumb held without exception. Let's define (a slightly technical notion of) an ARGUMENT as a series of sentences. The sentences at the beginning of the series are premises. The final sentence in the series is the conclusion.

Here is an example of an argument:

> People get wet whenever it rains.
> It often rains in Vancouver.
> ∴ People often get wet in Vancouver.

The idea of an argument is that the premises are supposed to give you reason to accept the conclusion. If I'm trying to convince you that people often get wet in Vancouver — the conclusion of the argument above — convincing you of the two premises might be a good way to get you there.

Notice that our definition of an argument is quite general. Consider this example:

> Vancouver has more churches than any other Canadian city.
> Two oboes are fighting a duel under the fireworks.
> ∴ J. Edgar Hoover was an honest man.

It may seem odd to call this an argument, but that is because it would be a terrible argument. The two premises have nothing at all to do with the conclusion. (Moreover, they aren't very plausible.) Nevertheless, given our definition, it still counts as an argument — albeit a bad one. One of our central aims in formal logic is to provide rigorous, formal tests for evaluating arguments.

1.2 Sentences and propositions

The premises and conclusions of arguments are sentences. But not just any English sentence is suitable for figuring into an argument. For example, questions count as grammatical sentences of English, but logical arguments never have questions as premises or conclusions. We are interested especially in sentences that can be true or false. Think of these as the sentences that purport to describe the way things are. Such sentences are sometimes described as expressing *propositions.*

We will not be theorizing in terms of questions and other non-propositional sentences in formal logic. Since we are only interested in sentences that can figure as a premise or conclusion of an argument, we'll define a technical notion of a SENTENCE as a sentence expressing a proposition, i.e., a sentence that can be true or false.

Don't confuse the idea of a sentence that can be true or false with the difference between fact and opinion. Often, sentences in logic will express things that would count as facts — such as 'Kierkegaard was a hunchback' or 'Kierkegaard liked almonds.' They can also express things that you might think of as matters of opinion — such as, 'Almonds are yummy' or 'the U.S. invasion of Iraq was unjustified'. These are all examples of things that are either true or false.

It is also important to keep clear the distinction between something's being *true* and something's being *known.* A sentence is the kind of thing that can be true or false; that doesn't mean you'll always be able to tell whether it is true or false. For example, 'there are an even number of humans on Earth right now' is a sentence. It is either true or false, even though it is pretty much impossible to tell which. Similarly, there are controversial propositions, where people disagree about whether they are true or false, and where it seems very difficult to settle the debate. (I think that 'Freedom of expression requires restricting oppressive speech' is an example.) This is either true or false, even if debates about which it is aren't likely to resolve the question. So it counts as a propositional sentence.

What are some examples of grammatical English sentences that do not express propositions? We've discussed one category already:

Questions   In a grammar class, 'Are you sleepy yet?' would count as an interrogative sentence. Although you might be sleepy or you might be alert, the question itself is neither true nor false. So 'Are you sleepy yet?' is not a sentence in our technical, propositional sense. Suppose you answer the question: 'I am not sleepy.' This *is* either true or false, and so it *is* a sentence in the logical sense. Generally, *questions* will not count as sentences, but *answers* will.

'What is this course about?' is not a sentence. 'No one knows what this course is about' is a sentence.

Imperatives   Commands are often phrased as imperatives like 'Wake up!', 'Sit up straight', and so on. In a grammar class, these would count as imperative sentences. Although it might be good for you to sit up straight or it might not, the command is neither true nor false. Note, however, that commands are not always phrased as imperatives. 'You will respect my authority' *is* either true or false — either you will or you will not — and so it counts as a sentence in the logical sense.

Exclamations   Expressions like 'Ouch!' or 'Boo, Yankees!' are sometimes described as exclamatory sentences, but they are neither true nor false. We will treat 'Ouch, I hurt my toe!' as meaning the same thing as 'I hurt my toe.' The 'ouch' does not add anything that could be true or false.

To recap: *sentences*, in the technical sense we're interested in, are claims that can be true or false. One pretty good test you can run to see whether something is a sentence is to ask whether it makes sense to insert 'it is true that' or 'it is false that' in front of it. It's perfectly fine to say 'it is true that Kierkegaard liked almonds' or 'it is true that the U.S. invasion of Iraq was unjustified'. But it doesn't make sense to say 'it is true that are you sleepy yet' or 'it is true that sit up straight'.

We will call truth or falsity the TRUTH-VALUE of a sentence.

1.3 Two ways that arguments can go wrong

Consider the argument that you should take an umbrella (on p. , above).

(1) It is raining heavily.
(2) When it rains, everyone outside without an umbrella gets wet.
∴ You should take an umbrella.

If premise (1) is false — if it is sunny outside — then the argument fails. It does not establish that you should take an umbrella. Or suppose that premise (2) is false. Maybe not everyone without an umbrella gets wet. (Maybe some people are able to stay under the awnings.) In this case, too, the argument does not establish its conclusion. Arguments only succeed when all their premises are true.

Suppose both premises *are* true. It really is raining heavily, and we're talking about an area where there's no protection from the rain other than umbrellas, so that anyone outside without an umbrella will get wet when it rains. Now does the argument show you that you should take an umbrella? It certainly seems to lend some support in that direction. But notice that it's still not *conclusive*. There are still *possible* ways the conclusion might be false. For example, suppose you like getting wet. Then, even if it is raining heavily, and even if everyone who goes outside without an umbrella when it rains gets wet, maybe you have no reason to take the umbrella.

A good argument — or at least, an argument that is good in one particularly interesting way — *compels* its conclusion.

So it seems that there are two ways an argument could be weak. First, one or more of the premises might be false. An argument gives you a reason to believe its conclusion only if you believe its premises. Second, the premises might fail to support the conclusion, or fail to support it sufficiently strongly. Even if the premises were true, the form of the argument might be weak. The example we just considered is weak in both ways.

Once again, we are working with a particular kind of idealization regarding arguments and what makes them good. Consider another example:

> You are reading this book.
> This is a logic book.
> ∴ You are a logic student.

In a straightforward and ordinary sense, this is not a terrible argument. The premises are true. And they do offer some support for the conclusion. Most people who read this book are logic students. Still, it is *possible* for someone besides a logic student to read this book. If the deans carefully read this book in order to find out whether I'm criticizing the UBC administration, that wouldn't make them logic students. So the premises of this argument, even though they are true, do not guarantee the truth of the conclusion.

By contrast, the sample argument about snow *is* valid. There's just no possible way for all three of its premises to be true without it snowing.

In logic, we are interested in arguments whose premises *guarantee* their conclusions. We call such arguments 'deductively valid' or just 'valid.' Even though we might count the rain argument, or the logic book argument, as a good argument in some sense, they are not valid; that is, they are 'invalid.' One important task of logic is to sort valid arguments from invalid arguments.

1.4 Validity

An argument is deductively VALID if and only if it is impossible for all the premises to be true while the conclusion is false. (We'll say a bit more about just what we mean by 'impossible' in §1.5 below.)

That's our official *definition* of validity. Once again (get used to this pattern), our formal definition is related to, but not quite the same as, the ordinary colloquial notion with the same name. In ordinary English, to say something is 'valid' means, somewhat vaguely, that it is good. (Think of a 'valid point' or a 'valid perspective'.) In logic, a valid argument is one that has this very specific feature: it's impossible for the premises to be true if the conclusion is false. (Side note: this is a good terminological distinction to keep in mind when writing essays in philosophy courses. Philosophy professors tend to wince and get annoyed when students use the word 'valid' in its imprecise sense. You can help stay on their good sides by only saying something is 'valid' when you mean that it is an argument whose premises are inconsistent with the falsity of its conclusion.)

Notice that validity does *not* require that the premises be true. Consider this example:

(1) Oranges are either fruits or musical instruments.
(2) Oranges are not fruits.
∴ Oranges are musical instruments.

This is a valid argument. It is impossible for the premises to be true if the conclusion is false. Since it has a false premise — premise (2) — it does not actually establish its conclusion, but it does have a *valid logical form. If* both premises were true, *then* the conclusion would necessarily be true.

Since this is a valid argument that is, in some important sense, a bad argument, this shows that validity isn't the only feature we care about in arguments. An argument is SOUND if and only if it is valid and all of its premises are true. For example, this variant argument on the above argument is sound:

(1) Oranges are either fruits or musical instruments.
(2) Oranges are not musical instruments.
∴ Oranges are fruits.

For reasons that will emerge, logic tends to focus primarily on validity, rather than soundness.

We've seen that a valid argument does not need to have true premises or a true conclusion. Conversely, having true premises and a true conclusion is also not enough to make an argument valid. Consider this example:

> Donald Trump is a U.S. citizen.
> Justin Trudeau is a Canadian citizen.
> ∴ UBC is the largest employer in Vancouver.

The premises and conclusion of this argument are, as a matter of fact, all true. Nevertheless, this is quite a poor argument. This is related to the fact that the premises have nothing to do with the conclusion. More precisely, the definition of validity is not satisfied: it is possible for the premises to be true while the conclusion is false. Although the conclusion is *actually* true, it is *possible* for the premises of this argument to be true while the conclusion is false. We may imagine, for example, that Lululemon hired an additional 100,000 employees in Vancouver, while UBC remained the same size, and without any world leaders changing their citizenship statuses. In such a possible scenario, the premises would be true, while the conclusion is false.

The important thing to remember is that validity is not about the actual truth or falsity of the sentences in the argument. Instead, it is about the form of the argument: The truth of the premises is incompatible with the falsity of the conclusion.

Here are some more valid arguments. Can you see why each is valid?

1. Socrates is a man.
2. All men are carrots.
∴ Socrates is a carrot.

1. Abe Lincoln was either born in Illinois or he was once president.
2. Abe Lincoln was never president.
∴ Abe Lincoln was born in Illinois.

1. Justin Trudeau is either from France or from Luxemborg.
2. Justin Trudeau is not from Luxemborg.
∴ Justin Trudeau is from France.

1. If the world were to end today, then I would not need to get up tomorrow morning.
2. I will need to get up tomorrow morning.
∴ The world will not end today.

1.5 Impossibility

The definition of validity centrally invokes the notion of impossibility. What it means for a sentence to be *impossible* is that there is no possible way for it to

be true. Impossible sentences don't just *happen* to be false; they *must* be false. Consider the sentence, 'The Whitecaps lost their last game.' As it happens, at the time I'm writing, that's false. They actually won. But it's not impossible; they might have lost, if a few things had gone differently. So that sentence is false, but not impossible.

By contrast, blatant contradictions like 'the Whitecaps won the game and the Whitecaps lost the game' *are* impossible. This sentence doesn't just *happen* to be false. It couldn't *possibly* be true.

Even falsehoods that are really obviously false, and for which it's hard to imagine how they could be true, might be *possible*. It's not *impossible* that pigs could fly. Someone might invent powerful wings for them. When we're interested in impossible sentences in logic, we really mean *impossible*. The best examples are contradictions.

So to apply the definition of validity, we consider whether there is any possibility, no matter how far-fetched, where the premises are true and the conclusion is false.

1.6 Other logical notions

Here are a few more relevant terms we'll be working with.

1.6.1 Logical truth

In considering arguments formally, we care about what would be true *if* the premises were true. Generally, we are not concerned with the actual truth value of any particular sentences — whether they are *actually* true or false. (As indicated above, we will be more interested in validity than in soundness.) Yet there are some sentences that must be true, just as a matter of logic.

Compare these sentences:

1. It is raining.
2. Either it is hot outside, or it is not hot outside.
3. There is an earthquake happening here right now and there are never earthquakes here.

Sentence 1 could be true or it could be false. (As I'm typing these words on a sunny Vancouver summer day, it is false.) Sentences that could be true, or could be false, are called *contingent* sentences.

Sentence 2 is different. Even though I don't know what the weather is like as you're reading this book, I still know that it's true. Sentence 2 *must* be true. This sentence is *logically true*; no matter how the weather or anything else happens to be, this sentence has to be true. We call a sentence like this a LOGICAL TRUTH or a TAUTOLOGY.

You do not need to check the weather to know about sentence 3, either. It must be false, simply as a matter of logic. There could be an earthquake now, but if so, it couldn't be the case that there are never earthquakes here. Or there might never be any earthquakes, but if so, there couldn't be one now. The third sentence is LOGICALLY FALSE; it is false regardless of what the world is like. A logically false sentence is called a CONTRADICTION.

I said above that a contingent sentence could be true and it also could be false. We can also define contingency in terms of tautologies and contradictions thus: a CONTINGENT SENTENCE is a sentence that is neither a tautology nor a contradiction.

1.6.2 Logical equivalence

We can also ask about the logical relations *between* two sentences. For example:

> Sunil went to the store after he washed the dishes.
> Sunil washed the dishes before he went to the store.

These two sentences are both contingent. (Do you see why?) Yet they must have the same truth-value. If either of the sentences is true, then they both are; if either of the sentences is false, then they both are. When two sentences necessarily have the same truth value, we say that they are LOGICALLY EQUIVALENT.

Notice that both of these arguments are valid:

> Sunil went to the store after he washed the dishes.
> ∴ Sunil washed the dishes before he went to the store.

> Sunil washed the dishes before he went to the store.
> ∴ Sunil went to the store after he washed the dishes.

In general, if two sentences are equivalent, then an argument with either one as a premise, and the other as the conclusion, will be valid.

1.6.3 Consistency

Consider these two sentences:

B1 My only brother is taller than I am.
B2 My only brother is shorter than I am.

Logic alone cannot tell us which, if either, of these sentences is true. Yet we can say that *if* the first sentence (B1) is true, *then* the second sentence (B2) must be false. And if B2 is true, then B1 must be false. It cannot be the case that both of these sentences are true.

If a set of sentences could not all be true at the same time, like B1–B2, they are said to be INCONSISTENT. Otherwise, they are CONSISTENT.

We can ask about the consistency of any number of sentences. For example, consider the following list of sentences:

G1 There are at least four giraffes at the wild animal park.
G2 There are exactly seven gorillas at the wild animal park.
G3 There are not more than two Martians at the wild animal park.
G4 Every giraffe at the wild animal park is a Martian.

G1 and G4 together imply that there are at least four Martian giraffes at the park. This conflicts with G3, which implies that there are no more than two Martian giraffes there. So the set of sentences G1–G4 is inconsistent. (Notice that in this example the inconsistency has nothing at all to do with G2. G2 just happens to be part of an inconsistent set.)

1.7 Formal languages

So far, we have been working with arguments expressed in English. But there are some general argumentative structures that transcend English or any other particular natural language. Consider, for example, the following simple argument:

Vancouver is in Canada.
Moscow is in Russia.
∴ Vancouver is in Canada and Moscow is in Russia.

Hopefully by now it is obvious to you that this argument is valid. (Not just that — it is even sound.) That argument has something important in common with this one:

> Vancouver is on Mars.
> Moscow is on Venus.
.˙. Vancouver is on Mars and Moscow is on Venus.

Although this one has false premises, and so can't be sound, it is also valid. It is impossible for the premises to be true if the conclusion is false. Not only that — it seems that the argument is valid in, in a sense to be articulated, *exactly the same way*. The validity of the first argument is very similar to that of the second. Even though they are expressed by different strings of English sentences, it is useful to develop a language in which they are, in an important sense, the same kind of argument. Over the course of this textbook, we'll be developing and working with such a language. We'll begin with a simple version of our logical language, called SL, for *sentential logic*. Later on, we'll develop a more complicated logical language, QL, for *quantified logic*. Both languages are *formal* in the sense that there are precise rules that govern how we should evaluate sentences within them.

English is a natural language, not a formal one. Its rules are vague and messy, and constantly changing. We will spend some time translating between English and our formal languages, but the translations will not always be precise. There is a tension between wanting to capture as much of the structure of English as possible and wanting a simple formal language with tractable rules — simpler formal languages will approximate natural languages less closely. There is no perfect formal language. Some will do a better job than others in translating particular English-language arguments.

In this book, we make the assumption that *true* and *false* are the only possible truth-values. Logical languages that make this assumption are called *bivalent*, which means *two-valued*. SL and QL are both bivalent, but some philosophers have emphasized limits to the power of bivalent logic. Some logics, beyond the scope of this book, allow for sentences that are neither true nor false. Others allow for sentences that are both true *and* false. Our logical system, which is often called *classical logic*, will give every sentence exactly one truth value: either true or false.

Summary of logical notions

> ▷ An ARGUMENT is a structured set of sentences, with *premises* intended to offer support for a *conclusion*.

> ▷ A SENTENCE, in our logical terminology, is a sentence that expresses a proposition, and can be either true or false.

> ▷ An argument is (deductively) VALID if it is impossible for the premises to be true and the conclusion false; it is INVALID otherwise.

> ▷ A TAUTOLOGY is a sentence that must be true, as a matter of logic.

> ▷ A CONTRADICTION is a sentence that must be false, as a matter of logic.

> ▷ A CONTINGENT SENTENCE is neither a tautology nor a contradiction.

> ▷ Two sentences are LOGICALLY EQUIVALENT if they necessarily have the same truth value.

> ▷ A set of sentences is CONSISTENT if it is logically possible for all the members of the set to be true at the same time; it is INCONSISTENT otherwise.

Practice Exercises

At the end of each chapter, you will find a series of practice problems that review and explore the material covered in the chapter. There is no substitute for actually working through some problems, because logic is more about a way of thinking than it is about memorizing facts. The answers to some of the problems are provided at the end of the book in appendix B; the problems that are solved in the appendix are marked with a ⋆.

⋆ **Part A** Which of the following are 'sentences' in the logical sense?

1. England is smaller than China.
2. Greenland is south of Jerusalem.
3. Is New Jersey east of Wisconsin?
4. The atomic number of helium is 2.
5. The atomic number of helium is π.
6. I hate overcooked noodles.
7. Blech! Overcooked noodles!
8. Overcooked noodles are disgusting.
9. Take your time.
10. This is the last question.

Part B For each of the following: Is it a tautology, a contradiction, or a contingent sentence?

1. Caesar crossed the Rubicon.
2. Someone once crossed the Rubicon.
3. No one has ever crossed the Rubicon.

4. If Caesar crossed the Rubicon, then someone has.
5. Even though Caesar crossed the Rubicon, no one has ever crossed the Rubicon.
6. If anyone has ever crossed the Rubicon, it was Caesar.

★ **Part C** Look back at the sentences G1–G4 on p. 19, and consider each of the following sets of sentences. Which are consistent? Which are inconsistent?

1. G2, G3, and G4
2. G1, G3, and G4
3. G1, G2, and G4
4. G1, G2, and G3

★ **Part D** Which of the following is possible? If it is possible, give an example. If it is not possible, explain why.

1. A valid argument that has one false premise and one true premise
2. A valid argument that has a false conclusion
3. A valid argument, the conclusion of which is a contradiction
4. An invalid argument, the conclusion of which is a tautology
5. A tautology that is contingent
6. Two logically equivalent sentences, both of which are tautologies
7. Two logically equivalent sentences, one of which is a tautology and one of which is contingent
8. Two logically equivalent sentences that together are an inconsistent set
9. A consistent set of sentences that contains a contradiction
10. An inconsistent set of sentences that contains a tautology

Part E Is this argument valid? Why or why not? (Hint: here and elsewhere in logic, read the definitions of our formal terms literally.)

(1) PHIL 220 is a course with a final exam.
(2) No courses ever have final exams.
∴ Everyone is going to get an A+ in PHIL 220.

Chapter 2

Sentential logic

This chapter introduces a logical language called SL. It is a version of *sentential logic*, because the basic units of the language will represent entire sentences. (Recall from §1.2 that we're only considering propositional sentences.)

2.1 Sentence letters

In SL, capital Roman letters (A, B, C, etc.) are used to represent basic sentences. Considered only as a symbol of SL, the letter A could mean any sentence. So when translating from English into SL, it is important to provide a *symbolization key*. The key provides an English language sentence for each sentence letter used in the symbolization.

For example, consider this argument:

> Today is New Year's Day.
> If today is New Year's Day, then people are swimming in English Bay.
> ∴ People are swimming in English Bay.

This is obviously a valid argument in English. In symbolizing it, we want to preserve the structure of the argument that makes it valid. What happens if we replace each sentence with a letter? Our symbolization key would look like this:

> **A:** Today is New Year's Day.
> **B:** If today is New Year's Day, then people are swimming in English Bay.
> **C:** People are swimming in English Bay.

We could then symbolize the argument in this way:

> *A*
> *B*
> ∴ *C*

This is a possible way to symbolize this argument, but it's not a very interesting one. There is no necessary connection between some sentence *A*, which could be any sentence, and some other sentences *B* and *C*, which could be any sentences. Something important about the argument has been lost in translation. The original argument was valid, but this translation of the argument does not reflect that validity. Given a different symbolization key, for example, the same argument form

> *A*
> *B*
> ∴ *C*

could equally well stand in for this invalid argument:

> Today is Christmas Day.
> Tiny Tim has difficulty walking without crutches.
> ∴ We're all going to die tomorrow.

A more interesting translation of the valid New Year's argument will show how it is different from the invalid Christmas argument. The relevant thing about the New Year's argument is that the second premise is not just *any* sentence. Notice that the second premise contains the first premise and the conclusion *as parts*. Our symbolization key for the argument only needs to include meanings for *A* and *C*, and we can build the second premise from those pieces. So we symbolize the argument this way:

> *A*
> If *A*, then *C*.
> ∴ *C*

This preserves the structure of the argument that makes it valid, but it still makes use of the English expression 'If. . . then. . . .' Although for our formal language we ultimately want to replace all of the English expressions with logical notation, this is a good start.

The sentences that can be symbolized with sentence letters are called *atomic sentences*, because they are the basic building blocks out of which more complex

sentences can be built. Whatever logical structure a sentence might have is lost when it is translated as an atomic sentence. From the point of view of SL, the sentence is just a letter. It can be used to build more complex sentences, but it cannot be taken apart.

We use capital Roman alphabet letters to represent SL sentences. There are only twenty-six such letters. We don't want to impose this artificial limit onto our formal language; it's better to work with a language that allows an arbitrary number of atomic sentences. To achieve this, we allow atomic sentences that have a capital letter with a numerical subscript. So we could have a symbolization key that looks like this:

A_1: Aang is from the Air Nation.
A_2: Aang is vegetarian.
A_3: Aang can bend water.
T_1: Toph is blind.
T_2: Toph likes badgers.
T_3: Toph invented metal bending.

Keep in mind that each of these is a different atomic sentence. Although it is often convenient, for making it easier to remember what each letter stands for, to use letters corresponding to the sentences' subject matters, as in the example above, no such requirement is built into the rules of SL. There is no special relationship between A_1 and A_2, as far as SL goes. It's just for our convenience that we might choose to make all the A sentences about Aang.

2.2 Connectives

Logical connectives are used to build complex sentences from atomic components. There are five logical connectives in SL. This table summarizes them. They are explained below.

symbol	what it is called	rough translation
¬	negation	'It is not the case that...'
&	conjunction	'Both... and ...'
∨	disjunction	'Either... or ...'
⊃	conditional	'If ... then ...'
≡	biconditional	'... if and only if ...'

Natural languages like English are vague and imprecise, and carry many complex subtleties of meaning. Our formal language, SL, has none of these properties. It is defined by precise rigid rules. Consequently, the 'translation' provided in the table is only an approximate one. We'll see some of the differences below.

2.3 Negation

Consider how we might symbolize these sentences:

1. Logic is hard.
2. It is false that logic is hard.
3. Logic isn't hard.

In order to symbolize sentence 1, we will need one sentence letter. We can provide a symbolization key:

   H: Logic is hard

Now, sentence 1 is simply H.

Since sentence 2 is obviously related to sentence 1, we do not want to introduce a different sentence letter. To put it partly in English, the sentence means 'It is false that H.' In order to symbolize this, we need a symbol for logical negation. We will use '¬.' Now we can translate 'Not H' to ¬H. In general, '¬' means 'it is false that'.

What of Sentence 3? It looks like a more natural way of saying the same thing as sentence 2. It's saying that logic isn't hard, which is just another way of denying that logic is hard. So sentences 2 and 3 have the same truth conditions. In the terminology of Ch. 1, they are logically equivalent. So we can translate both sentence 2 and sentence 3 as ¬H.

When translating from English into SL, the word 'not' is usually a pretty good clue that '¬' will be an appropriate symbol to use, but it's important to think about the actual meaning of the sentence, and not rely too much on which words appear in it.

> For any sentence Φ, a sentence can be symbolized as ¬Φ if it can be paraphrased in English as 'It is not the case that Φ.'

(For more on the 'Φ' notation, see the 'note about notation' on p. 30.)

Consider these further examples:

4. Rodrigo is mortal.
5. Rodrigo is immortal.

6. Rodrigo is not immortal.

If we let R mean 'Rodrigo is mortal', then sentence 4 can be translated as R.

What about sentence 5? Being immortal is pretty much the same as not being mortal. So it makes sense to treat 5 as the negation of 4, symbolizing it as $\neg R$.

Sentence 6 can be paraphrased as 'It is not the case that Rodrigo is immortal.' Using negation twice, we translate this as $\neg\neg R$. The two negations in a row each work as negations, so the sentence means 'It is not the case that... it is not the case that... R.' It is the negation of the negation of R. One can negate *any* sentence of SL by putting the ¬symbol in front of it. It's not only for atomic sentences.

Here is an example that illustrates some of the complexities of translation.

7. Elliott is happy.
8. Elliott is unhappy.

If we let H mean 'Elliot is happy', then we can symbolize sentence 7 as H.

We might be tempted to symbolize sentence 8 as $\neg H$. But is being unhappy the same thing as not being happy? This is perhaps debatable. One might think that it is possible to be neither happy nor unhappy. Maybe Elliot is in this in-between zone. If so, then we shouldn't treat 8 as the negation of 7. If we're allowing that 'unhappy' means something different from 'not happy', then we will need to use a different atomic sentence to translate 8.

What of the *truth conditions* for negated sentences?

For any sentence Φ: If Φ is true, then $\neg\Phi$ is false. If Φ is false, then $\neg\Phi$ is true. Using '1' for true and '0' for false, we can summarize this in a *characteristic truth table* for negation:

Φ	$\neg\Phi$
1	0
0	1

The left column shows the possible truth values for a given sentence; the right column shows the truth value of the negation of that sentence.

We will discuss truth tables at greater length in Chapter 3.

2.4 Conjunction

Consider these sentences:

9. Jessica is strong.
10. Luke is strong.
11. Jessica is strong, and Luke is also strong.

We will need separate sentence letters for 9 and 10, so we define this symbolization key:

J: Jessica is strong.
L: Luke is strong.

Sentence 9 can be symbolized as J.

Sentence 10 can be symbolized as L.

Sentence 11 can be paraphrased as 'J and L.' In order to fully symbolize this sentence, we need another symbol. We will use '$\&$.' We translate 'J and L' as $(J \& L)$. The logical connective '$\&$' is called CONJUNCTION, and J and L are each called CONJUNCTS.

Notice that we make no attempt to symbolize 'also' in sentence 11. Words like 'both' and 'also' function to draw our attention to the fact that two things are being conjoined. They are not doing any further logical work, so we do not need to represent them in SL. Note that Sentence 11 would have meant the same thing had it simply said 'Jessica is strong, and Luke is strong.'

Here are some more examples:

12. Jessica is strong and grumpy.
13. Jessica and Matt are both strong.
14. Although Luke is strong, he is not grumpy.
15. Matt is strong, but Jessica is stronger than Matt.

Sentence 12 is obviously a conjunction. The sentence says two things about Jessica, so in English it is permissible to use her name only once. It might be tempting to try this when translating the argument: Since J means 'Jessica is strong', one might attempt to paraphrase sentence 12 as 'J and grumpy.' But this would be a mistake. Once we translate part of a sentence as J, any further structure within the original sentence is lost. J is an atomic sentence; SL doesn't keep track of the fact that it was intended to be about Jessica. Moreover,

'grumpy' is not a sentence; on its own it is neither true nor false. So instead, we paraphrase sentence 12 as 'J and Jessica is grumpy.' Now we need to add a sentence letter to the symbolization key. Let G_1 mean 'Jessica is grumpy.' Now the sentence can be translated as $J \,\&\, G_1$.

> A sentence can be symbolized as $(\Phi \,\&\, \Psi)$ if it can be paraphrased in English as 'Both Φ, and Ψ.' Each of the conjuncts must be a sentence.

Sentence 13 says one thing about two different subjects. It says of both Jessica and Matt that they are strong, and in English we use the word 'strong' only once. In translating to SL, we want to make sure each conjunct is a sentence on its own, so once again, we'll paraphrase it by repeating the elements: 'Jessica is strong, and Matt is strong.' Once we add a new atomic sentence M for 'Matt is strong', this translates as $J \,\&\, M$.

Sentence 14 is a bit more complicated. The word 'although' tends to suggest a kind of contrast between the first part of the sentence and the second part. Nevertheless, the sentence is still telling us two things: Luke is strong, and he's not grumpy. So we can paraphrase sentence 14 as, '*Both* Luke is strong, *and* Luke is not grumpy.' The second conjunct contains a negation, so we paraphrase further: '*Both* Luke is strong *and it is not the case that* Luke is grumpy.' Let's let G_2 stand for 'Luke is grumpy', and we can translate sentence 14 as $L \,\&\, \neg G_2$.

Once again, this is an imperfect translation of the English sentence 14. That sentence implicated that there was a contrast between Luke's two properties. Our translation merely says that he has both of them. Still, it is a translation that preserves some of the important features of the original. In particular, it says that Luke is strong, and it also says that he's not grumpy.

Sentence 15's use of the word 'but' indicates a similar contrastive structure. It is irrelevant for the purpose of translating to SL, so we can paraphrase the sentence as '*Both* Matt is strong, *and* Jessica is stronger than Matt.' How should we translate the second conjunct? We already have the sentence letters J and M, which each say that one of Jessica and Matt is strong, but neither of these says anything comparative. We need a new sentence letter. Let S mean 'Jessica is stronger than Matt.' Now the sentence translates as $(M \,\&\, S)$.

> Sentences that can be paraphrased 'Φ, but Ψ' or 'Although Φ, Ψ' are best symbolized using conjunction: $(\Phi \,\&\, \Psi)$.

It is important to keep in mind that the sentence letters J, M, G_1, G_2, and S are atomic sentences. Considered as symbols of SL, they have no meaning beyond being true or false. We used J and M to symbolize different English

language sentences that are about people being strong, but this similarity is completely lost when we translate to SL. Nor does SL recognize any particular similarity between G_1 and G_2.

For any sentences Φ and Ψ, $(\Phi\,\&\,\Psi)$ is true if and only if both Φ and Ψ are true. We can summarize this in the characteristic truth table for conjunction:

Φ	Ψ	$(\Phi\,\&\,\Psi)$
1	1	1
1	0	0
0	1	0
0	0	0

The two left columns indicate the truth values of the two conjuncts. Since there are four possible combinations of truth values, there are four rows. The conjunction is true when both conjuncts are true, and false in all other cases.

2.4.1 A Note About Notation

In speaking generally about connectives, we've used the 'Φ' and 'Ψ' symbols as variables to stand in for sentences of SL. We saw, for instance, that for any sentence Φ, $\neg\Phi$ is the negation of Φ. This means:

▷ $\neg A$ is the negation of A

▷ $\neg B_1$ is the negation of B_1

▷ $\neg B_2$ is the negation of B_2

▷ $\neg\neg C$ is the negation of $\neg C$

▷ $\neg(A\,\&\,B)$ is the negation of $(A\,\&\,B)$

▷ etc.

Note that Φ and Ψ are *not* sentences of SL. They are symbols we use to talk about SL sentences, but they're not themselves part of our formal language. (Compare the use of variables like x in algebra. An 'x' symbol is used to stand in for any number, but x is not itself a number.) We'll return to this distinction in §2.9, and again later on when we discuss proving generalities about SL.

2.5 Disjunction

Consider these sentences:

16. Denison will golf with me or he will watch movies.
17. Either Denison or Ellery will golf with me.

For these sentences we can use this symbolization key:

D: Denison will golf with me.
E: Ellery will golf with me.
M: Denison will watch movies.

Sentence 16 is 'Either D or M.' To fully symbolize this, we introduce a new symbol. The sentence becomes $(D \lor M)$. The '$\lor$' connective is called DISJUNCTION, and D and M are called DISJUNCTS.

Sentence 17 is only slightly more complicated. There are two subjects, but the English sentence only gives the verb once. In translating, we can paraphrase it as 'Either Denison will play golf with me, or Ellery will play golf with me.' Now it obviously translates as $(D \lor E)$.

> A sentence can be symbolized as $(\Phi \lor \Psi)$ if it can be paraphrased in English as 'Either Φ, or Ψ.' Each of the disjuncts must be a sentence.

What truth conditions should we offer for '$\lor$' sentences? If I say, 'Denison will golf with me or he will watch movies', under what circumstances will that sentence be true? When will it be false? Well, suppose he doesn't do either activity. Suppose he goes swimming, and doesn't watch movies, and doesn't golf with me. Then my sentence is false.

Suppose Denison skips the movies and golfs with me instead. Then it seems pretty clear that my disjunctive claim was true. Likewise if he goes to the movies and leaves me without a golf partner. It's a little less clear what to think about the English sentence if *both* disjuncts end up true. Suppose that Denison comes golfing with me, *and also* stays up late afterward to come with me to the movies. Then is it true that 'he golfed with me or went to the movies'? This is not entirely clear. Certainly it would be strange to assert such a sentence if you know that both elements were true. On the other hand, it doesn't exactly seem *false* that he'll golf with me or watch movies, if in fact he'll do both. In a study of the semantics of English, it would be appropriate to pursue this question

much further. In this introduction to formal logic, we'll simply stipulate the features of our formal symbol. '∨' stands for an *inclusive or*, which means it is true if and only if *at least one disjunct* is true.

So $(D \lor E)$ is true if D is true, if E is true, or if both D and E are true. It is false only if both D and E are false. We can summarize this with the characteristic truth table for disjunction:

Φ	Ψ	$(\Phi \lor \Psi)$
1	1	1
1	0	1
0	1	1
0	0	0

Note that both conjunction and disjunction are symmetrical. $(\Phi \mathbin{\&} \Psi)$ is logically equivalent to $(\Psi \mathbin{\&} \Phi)$, and $(\Phi \lor \Psi)$ is logically equivalent to $(\Psi \lor \Phi)$.

These sentences are somewhat more complicated:

18. Either you will not have soup, or you will not have salad.
19. You will have neither soup nor salad.
20. You get soup or salad, but not both.

Here's a symbolization key:

S₁: You will get soup.
S₂: You will get salad.

Sentence 18 can be paraphrased in this way: 'Either *it is not the case that* you get soup, or *it is not the case that* you get salad.' Translating this requires both disjunction and negation. It becomes $(\neg S_1 \lor \neg S_2)$.

Sentence 19 also requires negation. It can be paraphrased as, '*It is not the case that* either that you get soup or that you get salad.' We need some way of indicating that the negation does not just negate the right or left disjunct, but rather negates the entire disjunction. In order to do this, we put parentheses around the disjunction: 'It is not the case that $(S_1 \lor S_2)$.' This becomes simply $\neg(S_1 \lor S_2)$. (A second, equivalent, way to translate this sentence is $(\neg S_1 \mathbin{\&} \neg S_2)$. We'll see why this is equivalent later on.)

Notice that the parentheses are doing important work here. The sentence $(\neg S_1 \lor S_2)$ would mean 'Either you will not have soup, or you will have salad,' which is very different.

Sentence 20 has a more complex structure. We can break it into two parts. The first part says that you get one or the other. We translate this as $(S_1 \lor S_2)$. The second part says that you do not get both. We can paraphrase this as, 'It is not the case both that you get soup and that you get salad.' Using both negation and conjunction, we translate this as $\neg(S_1 \,\&\, S_2)$. Now we just need to put the two parts together. As we saw above, 'but' can usually be translated as a conjunction. Sentence 20 can thus be translated as $((S_1 \lor S_2) \,\&\, \neg(S_1 \,\&\, S_2))$.

2.6 Conditional

For the following sentences, use this symbolization key:

R: You will cut the red wire.
B: The bomb will explode.

21. If you cut the red wire, then the bomb will explode.
22. The bomb will explode only if you cut the red wire.

Sentence 21 can be translated partially as 'If R, then B.' We will use the symbol '$\supset$' to represent this conditional relationship. The sentence becomes $(R \supset B)$. The connective is called a CONDITIONAL. The sentence on the left-hand side of the conditional (R in this example) is called the ANTECEDENT. The sentence on the right-hand side (B) is called the CONSEQUENT.

Sentence 22 is also a conditional. Since the word 'if' appears in the second half of the sentence, it might be tempting to symbolize this in the same way as sentence 21. That would be a mistake.

The conditional $(R \supset B)$ says that *if* R were true, *then* B would also be true. It does not say that your cutting the red wire is the *only* way that the bomb could explode. Someone else might cut the wire, or the bomb might be on a timer. The sentence $(R \supset B)$ does not say anything about what to expect if R is false. Sentence 22 is different. It says that the only conditions under which the bomb will explode are ones where you cut the red wire; i.e., if the bomb explodes, then you must have cut the wire. As such, sentence 22 should be symbolized as $(B \supset R)$.

It is important to remember that the connective '$\supset$' says only that, if the antecedent is true, then the consequent is true. It says nothing about the *causal* connection between the two events. Translating sentence 22 as $(B \supset R)$ does not mean that the bomb exploding would somehow have caused your cutting the wire. Both sentence 21 and 22 suggest that, if you cut the red wire, your

cutting the red wire would be the cause of the bomb exploding. They differ on the *logical* connection. If sentence 22 were true, then an explosion would tell us — those of us safely away from the bomb — that you had cut the red wire. Without an explosion, sentence 22 tells us nothing.

> The paraphrased sentence 'Φ only if Ψ' is logically equivalent to 'If Φ, then Ψ.'

'If Φ then Ψ' means that if Φ is true then so is Ψ. So we know that if the antecedent Φ is true but the consequent Ψ is false, then the conditional 'If Φ then Ψ' is false. What is the truth value of 'If Φ then Ψ' under other circumstances? Suppose, for instance, that the antecedent Φ happened to be false. 'If Φ then Ψ' would then not tell us anything about the actual truth value of the consequent Ψ, and it is unclear what the truth value of 'If Φ then Ψ' would be.

In English, the truth of conditionals often depends on what *would* be the case if the antecedent *were true* — even if, as a matter of fact, the antecedent is false. This poses a serious challenge for translating conditionals into SL. Considered as sentences of SL, R and B in the above examples have nothing intrinsic to do with each other. In order to consider what the world would be like if R were true, we would need to analyze what R says about the world. Since R is an atomic symbol of SL, however, there is no further structure to be analyzed. When we replace a sentence with a sentence letter, we consider it merely as some atomic sentence that might be true or false.

In order to translate conditionals into SL, we will not try to capture all the subtleties of English's 'if... then....' construction. Instead, the symbol '⊃' will signify a *material conditional*. This means that when Φ is false, the conditional $(\Phi \supset \Psi)$ is automatically true, regardless of the truth value of Ψ. If both Φ and Ψ are true, then the conditional $(\Phi \supset \Psi)$ is true.

In short, $(\Phi \supset \Psi)$ is false if and only if Φ is true and Ψ is false. We can summarize this with a characteristic truth table for the conditional.

Φ	Ψ	$\Phi \supset \Psi$
1	1	1
1	0	0
0	1	1
0	0	1

More than any other connective, the SL translation of the conditional is a rough approximation. It has some very counterintuitive consequences about the truth-values of conditionals. You can see from the truth table, for example, that an SL conditional is true any time the consequent is true, no matter what

the antecedent is. (Look at lines 1 and 3 in the chart.) And it is also true any time the antecedent is false, no matter what the consequent is. (Look at lines 3 and 4.) This is an odd consequence. In English, some conditionals with true consequents and/or false antecedents seem clearly to be false. For example:

23. If there are no philosophy courses at UBC, then PHIL 220 is a philosophy course at UBC.
24. If this book has fewer than thirty pages, then it will win the 2018 Pulitzer prize for poetry.

Both 23 and 24 seem clearly false. But each of them, translated into SL, would come out true. (If this isn't obvious, it's worth taking a moment to translate them and consider the truth table.) I told you before that English translations into SL are only approximate! Despite these odd results, the approach to conditionals offered here actually preserves many of the most important logical features of conditionals. We'll see this in more detail once we start working with proofs. For now, I'll just ask you to go along with this approach to conditionals, even though it will seem strange.

Note that unlike conjunction and disjunction, the conditional is *asymmetrical*. You cannot swap the antecedent and consequent without changing the meaning of the sentence, because ($\varPhi \supset \varPsi$) and ($\varPsi \supset \varPhi$) are not logically equivalent.

2.7 Biconditional

Consider these sentences:

25. The figure on the board is a triangle only if it has exactly three sides.
26. The figure on the board is a triangle if it has exactly three sides.
27. The figure on the board is a triangle if and only if it has exactly three sides.

    T: The figure is a triangle.
    S: The figure has three sides.

Sentence 25, for reasons discussed above, can be translated as ($T \supset S$).

Sentence 26 is importantly different. It can be paraphrased as, 'If the figure has three sides, then it is a triangle.' So it can be translated as ($S \supset T$).

Sentence 27 says that T is true *if and only if* S is true; we can infer S from T, and we can infer T from S. This is called a BICONDITIONAL, because it

entails the two conditionals $S \supset T$ and $T \supset S$. We will use '$\equiv$' to represent the biconditional; sentence 27 can be translated as $(S \equiv T)$.

We could abide without a new symbol for the biconditional. Since sentence 27 means '$(T \supset S)$ and $(S \supset T)$,' we could translate it as a conjunction of those two conditionals — as $((T \supset S) \& (S \supset T))$. Notice how the parentheses work: we need to add a new set of parentheses for the conjunction, in addition to the ones that were already there for the conditionals.

Because we could always write $((\Phi \supset \Psi) \& (\Psi \supset \Phi))$ instead of $(\Phi \equiv \Psi)$, we do not strictly speaking *need* to introduce a new symbol for the biconditional. Nevertheless, logical languages usually have such a symbol. SL will have one, which makes it easier to translate phrases like 'if and only if.'

$(\Phi \equiv \Psi)$ is true if and only if Φ and Ψ have the same truth value: they're either both true, or they're both false. This is the characteristic truth table for the biconditional:

Φ	Ψ	$\Phi \equiv \Psi$
1	1	1
1	0	0
0	1	0
0	0	1

2.8 Other symbolization

We have now introduced all of the connectives of SL. We can use them together to translate many kinds of sentences. Consider these examples of sentences that use the English-language connective 'unless', with an associated symbolization key:

28. Unless you wear a jacket, you will catch a cold.
29. You will catch a cold unless you wear a jacket.

   J: You will wear a jacket.
   D: You will catch a cold.

We can paraphrase sentence 28 as 'Unless J, D.' This means that if you do not wear a jacket, then you will catch a cold; with this in mind, we might translate it as $\neg J \supset D$. It also means that if you do not catch a cold, then you must have worn a jacket; with this in mind, we might translate it as $\neg D \supset J$.

Which of these is the correct translation of sentence 28? Both translations are correct, because the two translations are logically equivalent in SL.

Sentence 29, in English, is logically equivalent to sentence 28. It can be translated as either $\neg J \supset D$ or $\neg D \supset J$.

When symbolizing sentences like sentence 28 and sentence 29, it is easy to get turned around. Since the conditional is not symmetric, it would be wrong to translate either sentence as $J \supset \neg D$. Fortunately, there are other logically equivalent expressions. Both sentences mean that you will wear a jacket or — if you do not wear a jacket — then you will catch a cold. So we can translate them as $J \lor D$.

> If a sentence can be paraphrased as 'Unless Φ, Ψ,' then it can be symbolized as $(\neg\Phi \supset \Psi)$, $(\neg\Psi \supset \Phi)$, or $(\Phi \lor \Psi)$.

Symbolization of standard sentence types is summarized on p. 244.

2.9 Sentences of SL

The sentence 'Apples are red, or berries are blue' is a sentence of English, and the sentence '$(A \lor B)$' is a sentence of SL. Although we can identify sentences of English when we encounter them, we do not have a formal definition of 'sentence of English'. In SL, it is possible to formally define what counts as a sentence. This is one respect in which a formal language like SL is more precise than a natural language like English.

It is important to distinguish between the logical language SL, which we are developing, and the language that we use to talk about SL. When we talk about a language, the language that we are talking about is called the OBJECT LANGUAGE. The language that we use to talk about the object language is called the METALANGUAGE.

The object language in this chapter is SL. The metalanguage is English — not conversational English, but English supplemented with some logical and mathematical vocabulary (including the 'Φ' and 'Ψ' symbols). The sentence '$(A \lor B)$' is a sentence in the object language, because it uses only symbols of SL. The word 'sentence' is not itself part of SL, however, so the sentence 'This expression is a sentence of SL' is not a sentence of SL. It is a sentence in the metalanguage, a sentence that we use to talk *about* SL.

In this section, we will give a formal definition for 'sentence of SL.' The definition itself will be given in mathematical English, the metalanguage.

2.9.1 Expressions

There are three kinds of symbols in SL:

sentence letters	$A, B, C, \ldots, Z$
with subscripts, as needed	$A_1, B_1, Z_1, A_2, A_{25}, J_{375}, \ldots$
connectives	$\neg, \&, \vee, \supset, \equiv$
parentheses	(,)

We define an EXPRESSION OF SL as any string of symbols of SL. Take any of the symbols of SL and write them down, in any order, and you have an expression.

2.9.2 Well-formed formulae

Since any sequence of symbols is an expression, many expressions of SL will be gobbledegook. For example, these expressions don't mean anything:

$\neg\neg\neg\neg$

$))\equiv$

$A_4 \vee$

None of these are sentences in SL. A meaningful expression is called a WELL-FORMED FORMULA. It is common to use the acronym *wff*; the plural is wffs.

Individual sentence letters like A and G_{13} are certainly wffs. We can form further wffs out of these by using the various connectives. Using negation, we can get $\neg A$ and $\neg G_{13}$. Using conjunction, we can get $(A \& G_{13})$, $(G_{13} \& A)$, $(A \& A)$, and $(G_{13} \& G_{13})$. We could also apply negation repeatedly to get wffs like $\neg\neg A$ or apply negation along with conjunction to get wffs like $\neg(A \& G_{13})$ and $\neg(G_{13} \& \neg G_{13})$. The possible combinations are endless, even starting with just these two sentence letters, and there are infinitely many sentence letters. So there is no point in trying to list all the wffs.

Instead, we will describe the rules that govern how wffs can be constructed. Consider negation: Given any wff Φ of SL, $\neg\Phi$ is a wff of SL. Remember, Φ is not itself a sentence letter; it is a variable that stands in for any wff at all (atomic or not). Since the variable Φ is not a symbol of SL, $\neg\Phi$ is not an expression of SL. Instead, it is an expression of the metalanguage that allows us to talk about infinitely many expressions of SL: all of the expressions that start with the negation symbol. Because Φ is part of the metalanguage, it is called a *metavariable.*

Another way of saying that given any wff Φ of SL, $\neg\Phi$ is a wff of SL is to say that any time you have a wff of SL, you can make a new wff by adding a '$\neg$' symbol to the front of it.

We can say similar things for each of the other connectives. For instance, if Φ and Ψ are wffs of SL, then $(\Phi\,\&\,\Psi)$ is a wff of SL. (That is to say, given any two wffs in SL, you can put a '$\&$' symbol between them, a '(' in front of the first, and a ')' after the second, and the whole thing will add up to a new wff.) Providing clauses like this for all of the connectives, we arrive at the following formal definition for a well-formed formula of SL:

1. Every atomic sentence is a wff.

2. For all expressions Φ and Ψ,

   (a) If Φ is a wff, then $\neg\Phi$ is a wff.
   (b) If Φ and Ψ are wffs, then $(\Phi\,\&\,\Psi)$ is a wff.
   (c) If Φ and Ψ are wffs, then $(\Phi\vee\Psi)$ is a wff.
   (d) If Φ and Ψ are wffs, then $(\Phi\supset\Psi)$ is a wff.
   (e) If Φ and Ψ are wffs, then $(\Phi\equiv\Psi)$ is a wff.

3. Nothing else is a wff.

This is a *recursive* definition of a wff, illustrating how one can, starting with the simplest cases (atomic sentences), build up more complicated sentences of SL. If you have a wff, you can stick a '$\neg$' in front of it to make a new wff. If you have two wffs, you can put a '(' in front of the first one, followed by a '$\&$', followed by the second one, then finally a ')', and end up with a new wff. Etc.

Note that these are purely *syntactic* rules. They tell you how to construct an admissible ('grammatical') sentence in SL. They do not tell you what the sentence will *mean*. (We've touched on that already, in introducing characteristic truth tables. We'll return to this topic in much more detail in Ch. 3.)

Suppose we want to know whether or not $\neg\neg\neg D$ is a wff of SL. Looking at the second clause of the definition, we know that $\neg\neg\neg D$ is a wff *if* $\neg\neg D$ is a wff. So now we need to ask whether or not $\neg\neg D$ is a wff. Again looking at the second clause of the definition, $\neg\neg D$ is a wff *if* $\neg D$ is. Again, $\neg D$ is a wff *if* D is a wff. Now D is a sentence letter, an atomic sentence of SL, so we know that D is a wff by the first clause of the definition. So for a compound formula like $\neg\neg\neg D$, we must apply the definition repeatedly. Eventually we arrive at the atomic sentences from which the wff is built up.

The connective that you look to first in decomposing a sentence is called the MAIN CONNECTIVE (or *main logical operator*) of that sentence. For example:

The main connective of $\neg(E \vee (F \supset G))$ is negation, $\neg$. The main connective of $(\neg E \vee (F \supset G))$ is disjunction, $\vee$. Conversely, if you're building up a wff from simpler sentences, the connective introduced by the last rule you apply is the main connective. It is the connective that governs the interpretation of the entire sentence.

2.9.3 Sentences

Recall that a sentence is a meaningful expression that can be true or false. Since the meaningful expressions of SL are the wffs and since every wff of SL is either true or false, the definition for a sentence of SL is the same as the definition for a wff. Not every formal language will have this nice feature. In the language QL, which is developed later in the book, there are wffs which are not sentences.

The recursive structure of sentences in SL will be important when we consider the circumstances under which a particular sentence would be true or false. The sentence $\neg\neg\neg D$ is true if and only if the sentence $\neg\neg D$ is false, and so on through the structure of the sentence until we arrive at the atomic components: $\neg\neg\neg D$ is true if and only if the atomic sentence D is false. We will return to this point in much more detail in Chapters 3 and 4.

2.9.4 Notational conventions

A wff like $(Q \,\&\, R)$ must be surrounded by parentheses, because we might apply the definition again to use this as part of a more complicated sentence. If we negate $(Q \,\&\, R)$, we get $\neg(Q \,\&\, R)$. If we just had $Q \,\&\, R$ without the parentheses and put a negation in front of it, we would have $\neg Q \,\&\, R$. It is most natural to read this as meaning the same thing as $(\neg Q \,\&\, R)$, something very different than $\neg(Q \,\&\, R)$. The sentence $\neg(Q \,\&\, R)$ means that it is not the case that both Q and R are true; Q might be false or R might be false, but the sentence does not tell us which. The sentence $(\neg Q \,\&\, R)$ means specifically that Q is false and that R is true. So parentheses are crucial to the meaning of the sentence.

Consequently, strictly speaking, $Q \,\&\, R$ without parentheses is *not* a sentence of SL. When using SL, however, we will sometimes be able to relax the precise definition so as to make things easier for ourselves. We will do this in several ways.

First, we understand that $Q \,\&\, R$ means the same thing as $(Q \,\&\, R)$. As a matter of convention, we can leave off parentheses that occur *around the entire sentence.* Think of this as a kind of shorthand; we don't always write out the parentheses that we know are really there.

Second, it can sometimes be confusing to look at long sentences with nested sets of parentheses. We adopt the convention of using square brackets '[' and ']' in place of parenthesis. There is no logical difference between $(P \lor Q)$ and $[P \lor Q]$, for example. The unwieldy sentence

$$(((H \supset I) \lor (I \supset H)) \,\&\, (J \lor K))$$

could be written in this way, omitting the outer parentheses and using square brackets to make the inner structure easier to see:

$$\big[(H \supset I) \lor (I \supset H)\big] \,\&\, (J \lor K)$$

Third, we will sometimes want to translate the conjunction of three or more sentences. For the sentence 'Alice, Bob, and Candice all went to the party', suppose we let A mean 'Alice went', B mean 'Bob went', and C mean 'Candice went.' The definition only allows us to form a conjunction out of two sentences, so we can translate it as $(A \,\&\, B) \,\&\, C$ or as $A \,\&\, (B \,\&\, C)$. There is no reason to distinguish between these, since the two translations are logically equivalent. There is no logical difference between the first, in which $(A \,\&\, B)$ is conjoined with C, and the second, in which A is conjoined with $(B \,\&\, C)$. So we might as well just write $A \,\&\, B \,\&\, C$. As a matter of convention, we can leave out parentheses when we conjoin three or more sentences.

Fourth, a similar situation arises with multiple disjunctions. 'Either Alice, Bob, or Candice went to the party' can be translated as $(A \lor B) \lor C$ or as $A \lor (B \lor C)$. Since these two translations are logically equivalent, we may write $A \lor B \lor C$.

These latter two conventions only apply to multiple conjunctions or multiple disjunctions. If a series of connectives includes both disjunctions and conjunctions, then the parentheses are essential; as with $(A \,\&\, B) \lor C$ and $A \,\&\, (B \lor C)$. The parentheses are also required if there is a series of conditionals or biconditionals; as with $(A \supset B) \supset C$ and $A \equiv (B \equiv C)$.

We have adopted these four rules as *notational conventions*, not as changes to the definition of a sentence. Strictly speaking, $A \lor B \lor C$ is still not a sentence. Instead, it is a kind of shorthand. We write it for the sake of convenience, but we really mean the sentence $(A \lor (B \lor C))$.

Unless and until you are very confident about wffs and the use of parentheses, it is probably good advice to stick to the formal rules. These notational conventions are a way to skip steps when writing things down; if you're unsure about whether it's OK to take the shortcut, the safest thing is to go by the formal definition.

If we had given a different definition for a wff, then these could count as wffs. We might have written rule 3 in this way: "If Φ, Ψ, ... $\mathcal{Z}$ are wffs, then $(\Phi \,\&\, \Psi \,\&\, \ldots \,\&\, \mathcal{Z})$, is a wff." This would make it easier to translate some English sentences, but would have the cost of making our formal language more complicated. We would have to keep the complex definition in mind when we

develop truth tables and a proof system. We want a logical language that is *expressively simple* and allows us to translate easily from English, but we also want a *formally simple* language. (As we'll see later, this is important if we want to be able to prove things *about* our language.) Adopting notational conventions is a compromise between these two desires.

Practice Exercises

⋆ **Part A** Using the symbolization key given, translate each English-language sentence into SL.

> **M:** Those creatures are men in suits.
> **C:** Those creatures are chimpanzees.
> **G:** Those creatures are gorillas.

1. Those creatures are not men in suits.
2. Those creatures are men in suits, or they are not.
3. Those creatures are either gorillas or chimpanzees.
4. Those creatures are neither gorillas nor chimpanzees.
5. If those creatures are chimpanzees, then they are neither gorillas nor men in suits.
6. Unless those creatures are men in suits, they are either chimpanzees or they are gorillas.

Part B Using the symbolization key given, translate each English-language sentence into SL.

> **A:** Mister Ace was murdered.
> **B:** The butler did it.
> **C:** The cook did it.
> **D:** The Duchess is lying.
> **E:** Mister Edge was murdered.
> **F:** The murder weapon was a frying pan.

1. Either Mister Ace or Mister Edge was murdered.
2. If Mister Ace was murdered, then the cook did it.
3. If Mister Edge was murdered, then the cook did not do it.
4. Either the butler did it, or the Duchess is lying.
5. The cook did it only if the Duchess is lying.
6. If the murder weapon was a frying pan, then the culprit must have been the cook.

7. If the murder weapon was not a frying pan, then the culprit was either the cook or the butler.
8. Mister Ace was murdered if and only if Mister Edge was not murdered.
9. The Duchess is lying, unless it was Mister Edge who was murdered.
10. If Mister Ace was murdered, he was done in with a frying pan.
11. Since the cook did it, the butler did not.
12. Of course the Duchess is lying!

⋆ **Part C** Using the symbolization key given, translate each English-language sentence into SL.

E_1**:** Ava is an electrician.
E_2**:** Harrison is an electrician.
F_1**:** Ava is a firefighter.
F_2**:** Harrison is a firefighter.
S_1**:** Ava is satisfied with her career.
S_2**:** Harrison is satisfied with his career.

1. Ava and Harrison are both electricians.
2. If Ava is a firefighter, then she is satisfied with her career.
3. Ava is a firefighter, unless she is an electrician.
4. Harrison is an unsatisfied electrician.
5. Neither Ava nor Harrison is an electrician.
6. Both Ava and Harrison are electricians, but neither of them find it satisfying.
7. Harrison is satisfied only if he is a firefighter.
8. If Ava is not an electrician, then neither is Harrison, but if she is, then he is too.
9. Ava is satisfied with her career if and only if Harrison is not satisfied with his.
10. If Harrison is both an electrician and a firefighter, then he must be satisfied with his work.
11. It cannot be that Harrison is both an electrician and a firefighter.
12. Harrison and Ava are both firefighters if and only if neither of them is an electrician.

⋆ **Part D** Give a symbolization key and symbolize the following sentences in SL.

1. Alice and Bob are both spies.
2. If either Alice or Bob is a spy, then the code has been broken.
3. If neither Alice nor Bob is a spy, then the code remains unbroken.
4. The German embassy will be in an uproar, unless someone has broken the code.

5. Either the code has been broken or it has not, but the German embassy will be in an uproar regardless.
6. Either Alice or Bob is a spy, but not both.

⋆ **Part E** Give a symbolization key and symbolize the following sentences in SL.

1. If Gregor plays first base, then the team will lose.
2. The team will lose unless there is a miracle.
3. The team will either lose or it won't, but Gregor will play first base regardless.
4. Gregor's mom will bake cookies if and only if Gregor plays first base.
5. If there is a miracle, then Gregor's mom will not bake cookies.

Part F For each argument, write a symbolization key and translate the argument as well as possible into SL.

1. If Dorothy plays the piano in the morning, then Roger wakes up cranky. Dorothy plays piano in the morning unless she is distracted. So if Roger does not wake up cranky, then Dorothy must be distracted.
2. It will either rain or snow on Tuesday. If it rains, Neville will be sad. If it snows, Neville will be cold. Therefore, Neville will either be sad or cold on Tuesday.
3. If Zoog remembered to do his chores, then things are clean but not neat. If he forgot, then things are neat but not clean. Therefore, things are either neat or clean — but not both.

⋆ **Part G** For each of the following: (a) Is it, by the strictest formal standards, a sentence of SL? (b) Is it an acceptable way to write down a sentence of SL, allowing for our notational conventions?

1. (A)
2. $J_{374} \lor \neg J_{374}$
3. $\neg\neg\neg\neg F$
4. $\neg \,\&\, S$
5. $(G \,\&\, \neg G)$
6. $\Phi \supset \Phi$
7. $(A \supset (A \,\&\, \neg F)) \lor (D \equiv E)$
8. $[(Z \equiv S) \supset W] \,\&\, [J \lor X]$
9. $(F \equiv \neg D \supset J) \lor (C \,\&\, D)$

Part H

1. Are there any wffs of SL that contain no sentence letters? Why or why not?

Chapter 3

Truth tables

This chapter introduces a way of evaluating sentences and arguments of SL. The truth table method is a purely mechanical procedure that requires no intuition or special insight. Given the (albeit imprecise) translatability between SL and natural languages, this also amounts to a formal way of evaluating some natural language arguments.

3.1 Truth-functional connectives

Any non-atomic sentence of SL is composed of atomic sentences with sentential connectives. In Ch. 2, we offered *characteristic truth tables* for each connective. Although we didn't emphasize it at the time, the fact that it is possible to give truth tables like this is very significant. It means that our connectives are TRUTH-FUNCTIONAL. That is to say, the only thing that matters for determining the truth value of a given sentence of SL is the truth values of its constituent parts. To determine the truth value of a sentence $\neg\Phi$, the only thing that matters is the truth value of Φ. You don't have to know what Φ means, or where it came from, or what evidence there is for it and what that evidence might depend on. The truth-value of $\neg\Phi$ is a *function* of the truth-value of Φ. And so likewise for the other connectives.

This is a very interesting feature of SL. It is not inevitable. The syntax of English, for example, permits one to make a new, more complex English declarative sentence by prefixing the phrase 'Donald Trump cares whether' in front of any declarative English sentence. In this respect this phrase is syntactically similar to '$\neg$' in SL. But it is impossible to give a truth-functional characterization of the 'Donald Trump cares whether' operator in English. If you want to know

Φ	¬Φ
1	0
0	1

Φ	Ψ	Φ & Ψ	Φ∨Ψ	Φ⊃Ψ	Φ≡Ψ
1	1	1	1	1	1
1	0	0	1	0	0
0	1	0	1	1	0
0	0	0	0	1	1

Table 3.1: The characteristic truth tables for the connectives of SL.

whether Donald Trump cares whether the Canadian dollar is getting stronger, it's not enough to know whether the Canadian dollar is getting stronger. If it is, he might or might not care; if it isn't, he might or might not care. 'Donald Trump cares whether' is not truth-functional. But all the connectives of SL are.

For this reason, we can construct truth tables to determine the logical features of SL sentences.

3.2 Complete truth tables

The truth-value of sentences which contain only one connective are given by the characteristic truth table for that connective. Truth tables list out all the possible combinations of truth values for the atomic wffs at play; each row corresponds to an INTERPRETATION — a possible way of assigning truth values to atomic sentences. ($\{P = 1, Q = 0\}$ is an example of an interpretation.)

In the previous chapter, we wrote the characteristic truth tables with '1' for true and '0' for false. It is important to note, however, that this is not about truth in any deep or cosmic sense. Philosophical investigation into the nature of truth is a worthy project in its own right, but the truth functions in SL are just rules which transform input values into output values. (This is part of the reason why, in this book, we usually write '1' and '0' instead of 'T' and 'F'.) Even though we interpret '1' as meaning 'true' and '0' as meaning 'false', computers can be programmed to fill out truth tables in a purely mechanical way. In a machine, '1' might mean that a register is switched on and '0' that the register is switched off. Mathematically, they are just the two possible values that a sentence of SL can have. The truth tables for the connectives of SL, written in terms of 1s and 0s, are given in table 3.1.

The characteristic truth table for conjunction, for example, gives the truth conditions for any sentence of the form $(Φ \& Ψ)$. Even if the conjuncts $Φ$ and $Ψ$ are long, complicated sentences, the conjunction is true if and only if both $Φ$ and $Ψ$ are true.

Let's construct a truth table for a more complicated sentence. Consider the

sentence $(H \,\&\, I) \supset H$. We consider all the possible combinations of true and
false for H and I, which gives us four rows. We then copy the truth-values for
the sentence letters and write them underneath the letters in the sentence.

H	I	$(H$	$\&$	$I)$	$\supset$	H
1	1	**1**		**1**		**1**
1	0	**1**		**0**		**1**
0	1	**0**		**1**		**0**
0	0	**0**		**0**		**0**

So far all we've done is duplicate the first two columns. We've written the 'H'
column twice — once under each 'H', and the 'I' column once, under the 'I'.

Now consider the subsentence $H \,\&\, I$. This is a conjunction $\Phi \,\&\, \Psi$ with H as Φ
and with I as Ψ. H and I are both true on the first row. Since a conjunction
is true when both conjuncts are true, we write a 1 underneath the conjunction
symbol. In the other three rows, at least one of the conjuncts is false, so the
conjunction $(H \,\&\, I)$ is false. So we write 0s under the conjunction symbol on
those rows:

H	I	$(H$	$\&$	$I)$	$\supset$	H
		Φ	$\&$	Ψ		
1	1	1	**1**	1		1
1	0	1	**0**	0		1
0	1	0	**0**	1		0
0	0	0	**0**	0		0

The entire sentence is a conditional $\Phi \supset \Psi$ with $(H \,\&\, I)$ as Φ and with H as Ψ. On
the second row, for example, $(H \,\&\, I)$ is false and H is true. Since a conditional
is true when the antecedent is false, we write a 1 in the second row underneath
the conditional symbol. We continue for the other three rows and get this:

H	I	$(H$	$\&$	$I)$	$\supset$	H
			Φ		$\supset$	Ψ
1	1		1		**1**	1
1	0		0		**1**	1
0	1		0		**1**	0
0	0		0		**1**	0

The column of 1s underneath the conditional tells us that the sentence $(H \,\&\, I) \supset H$
is true regardless of the truth-values of H and I. They can be true or false in any
combination, and the compound sentence still comes out true. It is crucial that
we have considered all of the possible combinations. If we only had a two-line

truth table, we could not be sure that the sentence was not false for some other combination of truth-values.

In this example, we have not repeated all of the entries in every successive table, so that it's easier for you to see which parts are new. When actually writing truth tables on paper, however, it is impractical to erase whole columns or rewrite the whole table for every step. Although it is more crowded, the truth table can be written in this way:

H	I	(H & I) ⊃ H
1	1	1 1 1 **1** 1
1	0	1 0 0 **1** 1
0	1	0 0 1 **1** 0
0	0	0 0 0 **1** 0

Most of the columns underneath the sentence are only there for bookkeeping purposes. When you become more adept with truth tables, you will probably no longer need to copy over the columns for each of the sentence letters. In any case, the truth-value of the sentence on each row is just the column underneath the main logical operator of the sentence; in this case, the column underneath the conditional. We've marked it in bold.

A COMPLETE TRUTH TABLE has a row for all the possible combinations of 1 and 0 for all of the sentence letters. The size of the complete truth table depends on the number of different sentence letters in the table. A sentence that contains only one sentence letter requires only two rows, as in the characteristic truth table for negation. This is true even if the same letter is repeated many times, as in the sentence $[(C \equiv C) \supset C] \& \neg(C \supset C)$. The complete truth table requires only two lines because there are only two possibilities: C can be true or it can be false. A single sentence letter can never be marked both 1 and 0 on the same row. The truth table for this sentence looks like this:

C	[(C ≡ C) ⊃ C] & ¬ (C ⊃ C)
1	1 1 1 1 1 **0** 0 1 1 1
0	0 1 0 0 0 **0** 0 0 1 0

Looking at the column underneath the main connective, we see that the sentence is false on both rows of the table; i.e., it is false regardless of whether C is true or false. So it is a contradiction. (See §1.6.1.)

A sentence that contains two sentence letters requires four lines for a complete truth table, as in the other characteristic truth tables and the table for $(H \& I) \supset I$ above.

A sentence that contains three sentence letters requires eight lines. For example:

M	N	P	M & $(N \vee P)$				
1	1	1	1	**1**	1	1	1
1	1	0	1	**1**	1	1	0
1	0	1	1	**1**	0	1	1
1	0	0	1	**0**	0	0	0
0	1	1	0	**0**	1	1	1
0	1	0	0	**0**	1	1	0
0	0	1	0	**0**	0	1	1
0	0	0	0	**0**	0	0	0

From this table, we know that the sentence M & $(N \vee P)$ might be true or false, depending on the truth-values of M, N, and P.

A complete truth table for a sentence that contains four different sentence letters requires 16 lines. In general, if a complete truth table has n different sentence letters, then it must have 2^n rows.

In order to fill in the columns of a complete truth table, begin with the right-most sentence letter and alternate 1s and 0s. In the next column to the left, write two 1s, write two 0s, and repeat. For the third sentence letter, write four 1s followed by four 0s. This yields an eight line truth table like the one above. For a 16 line truth table, the next column of sentence letters should have eight 1s followed by eight 0s. For a 32 line table, the next column would have 16 1s followed by 16 0s. And so on.

3.3 Using truth tables

3.3.1 Tautologies, contradictions, and contingent sentences

Recall from §1.6.1 that an English sentence is a tautology if it must be true as a matter of logic. With a complete truth table, we consider all of the ways that the world might be. If the sentence is true on every line of a complete truth table, then it is true as a matter of logic, regardless of what the world is like.

So a sentence is a TAUTOLOGY IN SL if the column under its main connective is 1 on every row of a complete truth table.

Conversely, a sentence is a CONTRADICTION IN SL if the column under its main connective is 0 on every row of a complete truth table.

A sentence is CONTINGENT IN SL if it is neither a tautology nor a contradiction; i.e. if it is 1 on at least one row and 0 on at least one row.

From the truth tables in the previous section, we know that $(H$ & $I) \supset H$ is

a tautology, that $[(C \equiv C) \supset C] \& \neg(C \supset C)$ is a contradiction, and that $M \& (N \vee P)$ is contingent.

3.3.2 Logical equivalence

Two sentences are logically equivalent in English if they have the same truth value as a matter of logic. Once again, truth tables allow us to define an analogous concept for SL: Two sentences are LOGICALLY EQUIVALENT IN SL if they have the same truth-value on every row of a complete truth table.

Consider the sentences $\neg(A \vee B)$ and $\neg A \& \neg B$. Are they logically equivalent? To find out, we construct a truth table.

A	B	$\neg\,(A \vee B)$	$\neg\, A\, \&\, \neg\, B$
1	1	**0** 1 1 1	0 1 **0** 0 1
1	0	**0** 1 1 0	0 1 **0** 1 0
0	1	**0** 0 1 1	1 0 **0** 0 1
0	0	**1** 0 0 0	1 0 **1** 1 0

Look at the columns for the main connectives; negation for the first sentence, conjunction for the second. On the first three rows, both are 0. On the final row, both are 1. Since they match on every row, the two sentences are logically equivalent.

3.3.3 Consistency

A set of sentences in English is consistent if it is logically possible for them all to be true at once. A set of sentences is LOGICALLY CONSISTENT IN SL if there is at least one line of a complete truth table on which all of the sentences are true. It is INCONSISTENT otherwise.

Look again at the truth table above. We can see that $\neg(A \vee B)$ and $\neg A \& \neg B$ are consistent, because there is at least one row, namely the last one, where both are assigned true.

Are A, B, and $\neg A \& \neg B$ logically consistent? No. If they were, there would have to be a row where all three sentences are assigned true. But examination of the truth table reveals that there is no such row.

3.3.4 Validity

An argument in English is valid if it is logically impossible for the premises to be true and for the conclusion to be false at the same time. An argument is VALID IN SL if there is no row of a complete truth table on which the premises are all 1 and the conclusion is 0; an argument is INVALID IN SL if there is such a row.

Consider this argument:

$$\neg L \supset (J \vee L)$$
$$\neg L$$
$$\therefore J$$

Is it valid? To find out, we construct a truth table.

J	L	$\neg L \supset (J \vee L)$	$\neg L$	J
1	1	0 1 **1** 1 1 1	**0** 1	**1**
1	0	1 0 **1** 1 1 0	**1** 0	**1**
0	1	0 1 **1** 0 1 1	**0** 1	**0**
0	0	1 0 **0** 0 0 0	**1** 0	**0**

To determine whether the argument is valid, check to see whether there are any rows on which both premises are assigned 1, but where the conclusion is assigned 0. There are no such rows. The only row on which both the premises are 1 is the second row, and on that row the conclusion is also 1. So the argument form is valid in SL.

Here is another example. Is this argument valid?

$$P \supset Q$$
$$\neg P$$
$$\therefore \neg Q$$

To evaluate it, construct a truth table and see whether there is a row that assigns 1 to the premises and 0 to the conclusion:

P	Q	P	$\supset$	Q	$\neg$	P	$\neg$	Q
1	1	1	**1**	1	**0**	1	**0**	1
1	0	1	**0**	0	**0**	1	**1**	0
0	1	0	**1**	1	**1**	0	**0**	1
0	0	0	**1**	0	**1**	0	**1**	0

There is. On the third row, the premises are true and the conclusion is false. So the argument is invalid.

3.4 Partial truth tables

In order to show that a sentence is a tautology, we need to show that it is assigned 1 on every row. So we need a complete truth table. To show that a sentence is *not* a tautology, however, we only need one line: a line on which the sentence is 0. Therefore, in order to show that something is not a tautology, it is enough to provide a one-line *partial truth table* — regardless of how many sentence letters the sentence might have in it.

Consider, for example, the sentence $(U \& T) \supset (S \& W)$. We want to show that it is *not* a tautology by providing a partial truth table. We fill in 0 for the entire sentence. The main connective of the sentence is a conditional. In order for the conditional to be false, the antecedent must be true (1) and the consequent must be false (0). So we fill these in on the table:

S	T	U	W	$(U$ & $T)$ $\supset$ $(S$ & $W)$
				1 **0** 0

In order for the $(U \& T)$ to be true, both U and T must be true. So we put a 1 under those letters:

S	T	U	W	$(U$ & $T)$ $\supset$ $(S$ & $W)$
	1	1		1 1 1 **0** 0

Remember that each instance of a given sentence letter must be assigned the same truth value in a given row of a truth table. You can't assign 1 to one instance of U and 0 to another instance of U in the same row. So here we put a 1 under *each* instance of U and T.

Now we just need to make $(S \& W)$ false. To do this, we need to make at least one of S and W false. We can make both S and W false if we want. All that matters is that the whole sentence turns out false on this line. Making an arbitrary decision, we finish the table in this way:

S	T	U	W	$(U$ & $T)$ $\supset$ $(S$ & $W)$
0	1	1	0	1 1 1 **0** 0 0 0

Showing that something is a tautology requires a complete truth table. Showing that something is *not* a tautology requires only a one-line partial truth table, where the sentence is false on that one line. That's what we've just done. In the same way, to show that something is a contradiction, you must show that it is false on every row; to show that it is not a contradiction, you need only find one row where it is true.

A sentence is contingent if it is neither a tautology nor a contradiction. So showing that a sentence is contingent requires a *two-line* partial truth table: The sentence must be true on one line and false on the other. For example, we can show that the sentence above is contingent with this truth table:

S	T	U	W	$(U$ & $T) \supset (S$ & $W)$
0	1	1	0	1 1 1 **0** 0 0 0
0	1	0	0	0 0 1 **1** 0 0 0

Note that there are many combinations of truth values that would have made the sentence true, so there are many ways we could have written the second line.

Showing that a sentence is *not* contingent requires providing a complete truth table, because it requires showing that the sentence is a tautology or that it is a contradiction. If you do not know whether a particular sentence is contingent, then you do not know whether you will need a complete or partial truth table. You can always start working on a complete truth table. If you complete rows that show the sentence is contingent, then you can stop. If not, then complete the truth table. Even though two carefully selected rows will show that a contingent sentence is contingent, there is nothing wrong with filling in more rows.

Showing that two sentences are logically equivalent requires providing a complete truth table. Showing that two sentences are *not* logically equivalent requires only a one-line partial truth table: Make the table so that one sentence is true and the other false.

Showing that a set of sentences is consistent requires providing one row of a truth table on which all of the sentences are true. The rest of the table is irrelevant, so a one-line partial truth table will do. Showing that a set of sentences is inconsistent, on the other hand, requires a complete truth table: You must show that on every row of the table at least one of the sentences is false.

Showing that an argument is valid requires a complete truth table. Showing that an argument is *invalid* only requires providing a one-line truth table: If you can produce a line on which the premises are all true and the conclusion is false, then the argument is invalid.

Table 3.2 summarizes when a complete truth table is required and when a partial truth table will do. If you are trying to remember whether you need a complete truth table or not, the general rule is, if you're looking to establish a claim about *every* interpretation, you need a complete table.

	YES	NO
tautology?	complete truth table	one-line partial truth table
contradiction?	complete truth table	one-line partial truth table
contingent?	two-line partial truth table	complete truth table
equivalent?	complete truth table	one-line partial truth table
consistent?	one-line partial truth table	complete truth table
valid?	complete truth table	one-line partial truth table

Table 3.2: Do you need a complete truth table or a partial truth table? It depends on what you are trying to show.

3.5 Evaluating English arguments via SL

Recall from §1.4 that a natural language argument is valid if and only if it is impossible for the premises to be true while the conclusion is false. This notion is rather closely related to validity in SL, which obtains if and only if there is no assignment of truth values to atomic sentences on which the premises are assigned '1' and the conclusion is assigned '0'. This is of course by design. In many cases, validity in SL is a good indication of validity in English. Consider for example this English argument:

1. Either the butler is the murderer or the gardener isn't who he says he is.
2. The gardener is who he says he is.
∴ The butler is the murderer.

This argument is valid. There's no way for the conclusion to be false if both premises are true. Let's translate this argument into SL, and evaluate the resulting formal argument for validity with a truth table. We begin with a symbolization key:

B: The butler is the murderer.
G: The gardener is who he says he is.

With this key, the argument, rendered in SL, looks like this:

1. $(B \lor \neg G)$
2. G
∴ B

We can evaluate this formal argument for validity using a truth table. We'll set up a table with two atomic sentences, and check to see whether there is a

row that assigns '1' to both premises and assigns '0' to the conclusion. The completed truth table looks like this:

B	G	$(B \vee \neg G)$	G	B
1	1	1 **1** 0 1	**1**	**1**
1	0	1 **1** 1 0	**0**	**1**
0	1	0 **0** 0 1	**1**	**0**
0	0	0 **1** 1 0	**0**	**0**

If the argument form were invalid, there'd be a line on which the first two bold values are '1' but the third is '0'. There is no such line, so the argument form is valid. This helps explain why the English version of the argument is also valid: it is an argument that has a valid form in SL.

There are at least two advantages to evaluating natural language argument for validity by translating it into SL. For one thing, using truth tables to evaluate arguments is a formal method that does not require any particular rational insight or intuition. It is often relatively easy to tell whether the informal definition of validity is met — most of us have a good sense of what is and isn't possible. But it is an advantage to have an operationalized set of rules that a simple computer could apply. A second advantage, alluded to in Chapter 2, is that talking about valid SL forms provides a nice way to explain what various valid English arguments have in common with one another. The validity of the argument form

1. $(B \vee \neg G)$
2. G
∴ B

helps explain why the butler argument is valid, but the explanation does not depend on any of the specifics of what the individual letters stand for. It would be a perfectly good explanation for why *any* argument of this form would have to be valid. For example, consider this argument:

1. Either Barney is a purple dinosaur or I don't have a really weird-looking cat.
2. I do have a really weird-looking cat.
∴ Barney is a purple dinosaur.

That argument can be translated into the same SL argument just evaluated, using this symbolization key:

B: Barney is a purple dinosaur.

G: I have a really weird-looking cat.

Since it too has a valid form in SL, it too must be a valid argument. *Any* argument with this form will be a valid argument.

> If you have an argument form that is valid in SL, then *any* English argument that is properly translatable into that argument form will be valid.

Note that this is even true for arguments that have a more complex internal structure, like this strange argument:

1. Either Canada is a democracy if and only if Poland is neither part of the European Union nor majority Catholic, or my dog is not not lazy.
2. My dog is not lazy.
∴ Canada is a democracy if and only if Poland is neither part of the European Union nor majority Catholic.

One could translate this argument in to a relatively complex argument in SL. (If you did, the first premise would probably be a disjunction with a complex biconditional as one disjunct, and a negated negation as the other; the conclusion would probably be a biconditional with an atom on one side, and a negated disjunction on the other.) In this instance, however, that extra structure isn't needed to explain the validity of the argument. For this argument too has the same valid form as the previous ones, as can be seen via this symbolization key:

B: Canada is a democracy if and only if Poland is neither part of the European Union nor majority Catholic.
G: My dog is not lazy.

For many purposes, this would be a poor choice of symbolization keys. It ignores the internal structure of two central sentences. But if, as here, that structure is irrelevant, you can save yourself some work by engaging at a higher level of abstraction.

Note also that it's an implication of what we've just illustrated that there's not just one SL argument that is *the* argument form for a given argument in English. One can formalize arguments in various ways. If an argument has a valid argument form in SL, that's a guarantee that the argument is valid. But it is important not to invert this relationship — it does *not* follow from the fact that an argument has an *invalid* form, that the argument is invalid. One reason this is so is that it is possible for some arguments to have valid forms *and* invalid forms.

3.6 Common student mistakes

Here are a couple of issues that often come up when I teach this material to students.

3.6.1 Validity is not relative to particular interpretations

Truth in SL is relative to an interpretation. A given wff is true on some interpretations, and false on others. For example, $P \equiv Q$ is true when $\{P = 1, Q = 1\}$, and false when $\{P = 1, Q = 0\}$. But validity, unlike truth, is a claim about *all* interpretations.

Let's use a truth table to evaluate whether $P \vee Q$, $\neg P$ $\therefore$ $P \equiv Q$ is a valid argument form. To do this, we can draw the complete truth table for those three wffs:

P	Q	P	$\vee$	Q	$\neg$	P	P	$\equiv$	Q
1	1	1	**1**	1	**0**	1	1	**1**	1
1	0	1	**1**	0	**0**	1	1	**0**	0
0	1	0	**1**	1	**1**	0	0	**0**	1
0	0	0	**0**	0	**1**	0	0	**1**	0

We then check to see whether there is any row where the disjunction and the negation are true and the biconditional is false. Row three, $\{P = 0, Q = 1\}$, is such a row, so the argument is invalid.

Here is a common mistake. Students sometimes draw the truth table accurately, and correctly note that line three is inconsistent with the argument's validity, but misdescribe the situation, saying something like this: *'The argument is invalid when $P = 0$, $Q = 1$.'* This is a mistake because the argument isn't only invalid *in that row* — the argument in invalid, *period.* That interpretation *explains why* the argument is invalid, but validity and invalidity are claims about *all* interpretations. So we shouldn't say the argument is invalid 'when' $P = 0$, $Q = 1$; it's invalid always, because of the interpretation where $P = 0$, $Q = 1$.

This sounds like a small and fussy point about speaking precisely about logic. It is. But careful language here can help lead to careful thought. It is important, to thinking clearly about validity, to remember that validity, unlike truth, is a feature that arguments have or lack intrinsically. It is not something that an argument has in some interpretations and lacks in others.

The same goes for tautologies, contradictions, and contingent sentences. A sentence isn't a tautology on some interpretations and not on others; it either

simply is, or it simply is not, a tautology.

3.6.2 Valid arguments with invalid forms

Here is a question students sometimes get wrong: *Provide (in English) a valid argument of the form P $\therefore$ P & Q, or explain why it is impossible to do so.* Students often think this is impossible, because $P \therefore P \& Q$ is an invalid argument form. That certainly is an invalid argument form. The premise is true, and the conclusion false, when $P = 1$, $Q = 0$. But that doesn't mean there can be no valid argument with this form.

A valid argument form is a form that guarantees that any argument of that form is valid. Since this form is invalid, it is going to be possible to find invalid arguments of that form. For example, we could interpret P and Q thus:

> **P:** Captain Hook is angry.
> **Q:** Captain Hook has at least two daughters.

And of course the English argument, *Captain Hook is angry, therefore Captain Hook is angry and he has at least two daughters* is obviously invalid. But that doesn't mean there couldn't be *other* arguments of this form that *are* valid. The form doesn't guarantee that they'll be valid, but they may be valid nevertheless. Consider this symbolization key:

> **P:** Captain Hook has three daughters.
> **Q:** Captain Hook has at least two daughters.

The resulting argument is valid: *Captain Hook has three daughters. Therefore, Captain Hook has three daughters and he has at least two daughters.* It is impossible for the premise to be true without the conclusion also being true, so it satisfies the definition of validity given in Chapter 1. This argument is valid, even though it has an SL argument form that is invalid. We will be particularly interested in arguments whose validity can be explained by their forms, but the definition of validity in English is more general.

Practice Exercises

If you want additional practice, you can construct truth tables for any of the sentences and arguments in the exercises for the previous chapter.

⋆ **Part A** Determine whether each sentence is a tautology, a contradiction, or a contingent sentence. Justify your answer with a complete or partial truth table where appropriate.

1. $A \supset A$
2. $\neg B \,\&\, B$
3. $C \supset \neg C$
4. $\neg D \lor D$
5. $(A \equiv B) \equiv \neg(A \equiv \neg B)$
6. $(A \,\&\, B) \lor (B \,\&\, A)$
7. $(A \supset B) \lor (B \supset A)$
8. $\neg[A \supset (B \supset A)]$
9. $(A \,\&\, B) \supset (B \lor A)$
10. $A \equiv [A \supset (B \,\&\, \neg B)]$
11. $\neg(A \lor B) \equiv (\neg A \,\&\, \neg B)$
12. $\neg(A \,\&\, B) \equiv A$
13. $\big[(A \,\&\, B) \,\&\, \neg(A \,\&\, B)\big] \,\&\, C$
14. $A \supset (B \lor C)$
15. $[(A \,\&\, B) \,\&\, C] \supset B$
16. $(A \,\&\, \neg A) \supset (B \lor C)$
17. $\neg\big[(C \lor A) \lor B\big]$
18. $(B \,\&\, D) \equiv [A \equiv (A \lor C)]$

⋆ **Part B** Determine whether each pair of sentences is logically equivalent. Justify your answer with a complete or partial truth table where appropriate.

1. A, $\neg A$
2. A, $A \lor A$
3. $A \supset A$, $A \equiv A$
4. $A \lor \neg B$, $A \supset B$
5. $A \,\&\, \neg A$, $\neg B \equiv B$
6. $\neg(A \,\&\, B)$, $\neg A \lor \neg B$
7. $\neg(A \supset B)$, $\neg A \supset \neg B$
8. $(A \supset B)$, $(\neg B \supset \neg A)$
9. $[(A \lor B) \lor C]$, $[A \lor (B \lor C)]$
10. $[(A \lor B) \,\&\, C]$, $[A \lor (B \,\&\, C)]$

⋆ **Part C** Determine whether each set of sentences is consistent or inconsistent. Justify your answer with a complete or partial truth table where appropriate.

1. $A \supset A$, $\neg A \supset \neg A$, $A \,\&\, A$, $A \lor A$
2. $A \,\&\, B$, $C \supset \neg B$, C
3. $A \lor B$, $A \supset C$, $B \supset C$
4. $A \supset B$, $B \supset C$, A, $\neg C$

5. $B \mathbin{\&} (C \lor A)$, $A \supset B$, $\neg(B \lor C)$
6. $A \lor B$, $B \lor C$, $C \supset \neg A$
7. $A \equiv (B \lor C)$, $C \supset \neg A$, $A \supset \neg B$
8. A, B, C, $\neg D$, $\neg E$, F

⋆ **Part D** Determine whether each argument is valid or invalid. Justify your answer with a complete or partial truth table where appropriate.

1. $A \supset A$ ∴ A
2. $A \lor \big[A \supset (A \equiv A) \big]$ ∴ A
3. $A \supset (A \mathbin{\&} \neg A)$ ∴ $\neg A$
4. $A \equiv \neg(B \equiv A)$ ∴ A
5. $A \lor (B \supset A)$ ∴ $\neg A \supset \neg B$
6. $A \supset B$, B ∴ A
7. $A \lor B$, $B \lor C$, $\neg A$ ∴ $B \mathbin{\&} C$
8. $A \lor B$, $B \lor C$, $\neg B$ ∴ $A \mathbin{\&} C$
9. $(B \mathbin{\&} A) \supset C$, $(C \mathbin{\&} A) \supset B$ ∴ $(C \mathbin{\&} B) \supset A$
10. $A \equiv B$, $B \equiv C$ ∴ $A \equiv C$

⋆ **Part E** Answer each of the questions below and justify your answer.

1. Suppose that Φ and Ψ are logically equivalent. What can you say about $\Phi \equiv \Psi$?
2. Suppose that $(\Phi \mathbin{\&} \Psi) \supset \Omega$ is contingent. What can you say about the argument "Φ, Ψ, ∴Ω"?
3. Suppose that $\{\Phi, \Psi, \Omega\}$ is inconsistent. What can you say about $(\Phi \mathbin{\&} \Psi \mathbin{\&} \Omega)$?
4. Suppose that Φ is a contradiction. What can you say about the argument "Φ, Ψ, ∴Ω"?
5. Suppose that Ω is a tautology. What can you say about the argument "Φ, Ψ, ∴Ω"?
6. Suppose that Φ and Ψ are logically equivalent. What can you say about $(\Phi \lor \Psi)$?
7. Suppose that Φ and Ψ are *not* logically equivalent. What can you say about $(\Phi \lor \Psi)$?

Part F We could leave the biconditional ($\equiv$) out of the language. If we did that, we could still write '$A \equiv B$' so as to make sentences easier to read, but that would be shorthand for $(A \supset B) \mathbin{\&} (B \supset A)$. The resulting language would be formally equivalent to SL, since $A \equiv B$ and $(A \supset B) \mathbin{\&} (B \supset A)$ are logically equivalent in SL. If we valued formal simplicity over expressive richness, we could replace more of the connectives with notational conventions and still have a language equivalent to SL.

There are a number of equivalent languages with only two connectives. It would be enough to have only negation and the material conditional. Show this by writing sentences that are logically equivalent to each of the following using only parentheses, sentence letters, negation ($\neg$), and the material conditional ($\supset$).

⋆ 1. $A \vee B$
⋆ 2. $A \& B$
⋆ 3. $A \equiv B$

We could have a language that is equivalent to SL with only negation and disjunction as connectives. Show this: Using only parentheses, sentence letters, negation ($\neg$), and disjunction ($\vee$), write sentences that are logically equivalent to each of the following.

4. $A \& B$
5. $A \supset B$
6. $A \equiv B$

The *Sheffer stroke* is a logical connective with the following characteristic truthtable:

Φ	Ψ	$\Phi|\Psi$
1	1	0
1	0	1
0	1	1
0	0	1

7. Write a sentence using the connectives of SL that is logically equivalent to $(A|B)$.

Every sentence written using a connective of SL can be rewritten as a logically equivalent sentence using one or more Sheffer strokes. Using no connectives other than the Sheffer stroke, write sentences that are equivalent to each of the following.

8. $\neg A$
9. $(A \& B)$
10. $(A \vee B)$
11. $(A \supset B)$
12. $(A \equiv B)$

Chapter 4

Entailment and Models for SL

This chapter offers a formal semantics for SL, allowing us to be more precise about the notion of truth in SL. We'll also highlight some important features of *entailment*, a key concept in formal logic.

A formal, logical language is built from two kinds of elements: logical symbols and non-logical symbols. Connectives like ' & ' and ' ⊃ ' are logical symbols, because their meaning is specified within the formal language. When writing a symbolization key, you are not allowed to change the meaning of the logical symbols. You cannot say, for instance, that the '¬' symbol will mean 'not' in one argument and 'perhaps' in another. The '¬' symbol always means logical negation. It is used to translate the English language word 'not', but it is a symbol of a formal language and is defined by its truth conditions.

The sentence letters in SL are non-logical symbols, because their meaning is not defined by the logical structure of SL. When we translate an argument from English to SL, for example, the sentence letter M does not have its meaning fixed in advance; instead, we provide a symbolization key that says how M should be interpreted in that argument.

In translating from English to a formal language, we provided symbolization keys which were interpretations of all the non-logical symbols we used in the translation. An INTERPRETATION gives a meaning to all the non-logical elements of the language. We'll also use the term 'MODEL' as another word for an interpretation. In our simple formal language SL, meaning is simply a matter of truth and falsity, relative to a given interpretation. These notions too need to be characterized in a formal way.

4.1 Semantics for SL

This section provides a rigorous, formal characterization of *truth in SL* which builds on what we already know from doing truth tables. We were able to use truth tables to reliably test whether a sentence was a tautology in SL, whether two sentences were equivalent, whether an argument was valid, and so on. For instance: Φ is a tautology in SL if and only if it is assigned '1' on every line of a complete truth table.

This worked because each line of a truth table corresponds to a way the world might be. We considered all the possible combinations of 1 and 0 for the sentence letters that made a difference to the sentences we cared about. The truth table allowed us to determine what would happen given these different combinations. Each of those combinations of truth values is an *interpretation* in SL.

Once we construct a truth table, the symbols '1' and '0' are divorced from their metalinguistic meaning of 'true' and 'false'. We interpret '1' as meaning 'true', but the formal properties of 1 are defined by the characteristic truth tables for the various connectives. The symbols in a truth table have a formal meaning that we can specify entirely in terms of how the connectives operate. For example, if A is value 1, then $\neg A$ is value 0.

To formally define truth in SL, then, we want a function that assigns, for each model, a 1 or 0 to each of the sentences of SL. We can interpret this function as a definition of truth for SL if it assigns 1 to all of the true sentences of SL and 0 to all of the false sentences of SL. Call this function 'v' (for 'valuation'). We want v to be a function such that for any sentence Φ, $v(\Phi) = 1$ if Φ is true and $v(\Phi) = 0$ if Φ is false.

Recall that the recursive definition of a wff for SL had two stages: The first step said that atomic sentences (solitary sentence letters) are wffs. The second stage allowed for wffs to be constructed out of more basic wffs. There were clauses of the definition for all of the sentential connectives. For example, if Φ is a wff, then $\neg\Phi$ is a wff.

Our strategy for defining the truth function, v, will also be in two steps. The first step will handle truth for atomic sentences; the second step will handle truth for compound sentences.

4.2 Defining truth in SL

How can we define truth for an atomic sentence of SL? Consider, for example, the sentence M. Without an interpretation, we cannot say whether M is true or

false. It might mean anything. If we use M to symbolize 'The moon orbits the Earth', then M is true. If we use M to symbolize 'The moon is a giant turnip', then M is false.

When we give a symbolization key for SL, we provide an interpretation of the sentence letters that we use. The key gives an English language sentence for each sentence letter that we use. In this way, the interpretation specifies what each of the sentence letters *means.* However, this is not enough to determine whether or not that sentence is true. The sentences about the moon, for instance, require that you know some rudimentary astronomy. Imagine a small child who became convinced that the moon is a giant turnip. She could understand what the sentence 'The moon is a giant turnip' means, but mistakenly think that it was true.

So an interpretation alone does not determine whether a sentence is true or false. Truth or falsity depends also on what the world is like. If M meant 'The moon is a giant turnip' and the real moon were a giant turnip, then M would be true. To put the point in a general way, truth or falsity is determined by an interpretation *plus* a way that the world is.

INTERPRETATION + STATE OF THE WORLD $\implies$ TRUTH/FALSITY

In providing a logical definition of truth, we will not be able to give an account of how an atomic sentence is made true or false by the world. Instead, we will introduce a *truth value assignment.* Formally, this will be a function that tells us the truth value of all the atomic sentences. Call this function 'a' (for 'assignment'). We define a for all sentence letters $\mathcal{P}$, such that

$$a(\mathcal{P}) = \left\{ \begin{array}{ll} 1 & \text{if } \mathcal{P} \text{ is true,} \\ 0 & \text{otherwise.} \end{array} \right.$$

This means that a takes any atomic sentence of SL and assigns it either a one or a zero; one if the sentence is true, zero if the sentence is false.

You can think of a as being like a row of a truth table. Whereas a truth table row assigns a truth value to a few atomic sentences, the truth value assignment assigns a value to every atomic sentence of SL. There are infinitely many sentence letters, and the truth value assignment gives a value to each of them. When constructing a truth table, we only care about sentence letters that affect the truth value of sentences that interest us. As such, we ignore the rest. Strictly speaking, every row of a truth table gives a *partial* truth value assignment.

It is important to note that the truth value assignment, a, is not part of the language SL. Rather, it is part of the mathematical machinery that we are using to describe SL. It encodes which atomic sentences are true and which are false.

We now define the truth function, v, using the same recursive structure that we used to define a wff of SL.

1. If Φ is a sentence letter, then $v(\Phi) = a(\Phi)$.

2. If Φ is $\neg\Psi$ for some sentence Ψ, then

$$v(\Phi) = \begin{cases} 1 & \text{if } v(\Psi) = 0, \\ 0 & \text{otherwise.} \end{cases}$$

3. If Φ is $(\Psi \,\&\, \Omega)$ for some sentences Ψ, Ω, then

$$v(\Phi) = \begin{cases} 1 & \text{if } v(\Psi) = 1 \text{ and } v(\Omega) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

You may be tempted to worry that this definition is circular, because it uses the word 'and' in trying to define 'and.' But remember, we are not attempting to give a definition of the English word 'and'; we are giving a definition of truth for sentences of SL containing the logical symbol '$\&$.' We define truth for object language sentences containing the symbol '$\&$' using the metalanguage word 'and.' There is nothing circular about that.

4. If Φ is $(\Psi \vee \Omega)$ for some sentences Ψ, Ω, then

$$v(\Phi) = \begin{cases} 0 & \text{if } v(\Psi) = 0 \text{ and } v(\Omega) = 0, \\ 1 & \text{otherwise.} \end{cases}$$

5. If Φ is $(\Psi \supset \Omega)$ for some sentences Ψ, Ω, then

$$v(\Phi) = \begin{cases} 0 & \text{if } v(\Psi) = 1 \text{ and } v(\Omega) = 0, \\ 1 & \text{otherwise.} \end{cases}$$

6. If Φ is $(\Psi \equiv \Omega)$ for some sentences Ψ, Ω, then

$$v(\Phi) = \begin{cases} 1 & \text{if } v(\Psi) = v(\Omega), \\ 0 & \text{otherwise.} \end{cases}$$

Since the definition of v has the same structure as the definition of a wff, we know that v assigns a value to *every* wff of SL. Since the sentences of SL and the wffs of SL are the same, this means that v returns the truth value of every sentence of SL.

Truth in SL is always truth *relative to* some interpretation, because the definition of truth for SL does not say whether a given sentence is true or false. Rather, it says how the truth of that sentence relates to a truth value assignment.

4.3 Semantic entailment

We are now in a position to give more precise definitions of terms like 'tautology', 'contradiction', and so on. Truth tables provided a way to *check* whether a sentence was a tautology in SL, but they did not *define* what it means to be a tautology in SL. We will give definitions of these concepts for SL in terms of ENTAILMENT.

The relation of semantic entailment is about satisfiability — that is, whether there is any possible interpretation that meets a certain set of conditions. 'Φ entails Ψ', means that there is no interpretation for which Φ is true and Ψ is false. In SL, an interpretation is just an assignment of truth values to atomic sentences; so Φ entails Ψ if and only if every assignment of truth values to atomic sentences that makes Φ true also makes Ψ true.

We abbreviate entailment with a symbol called the *double turnstile*: $\Phi \models \Psi$ means 'Φ semantically entails Ψ.'

The double turnstile, like 'Φ', etc., is part of the *metalanguage* we use to discuss SL; it is not part of SL itself.

4.4 Entailment, validity, and informally good arguments

Entailment is a formal notion. It is connected in important ways to the informal notion of a good argument, but when considering entailment in SL, it is important to remember to apply the definitions rigorously and precisely, rather than relying on your sense of whether the argument is a good one. The notions can come apart in some surprising ways.

Let's start with a straightforward example. Consider whether this entailment claim is true:

$$(P \,\&\, Q) \models (P \lor Q)$$

This is true if and only if every interpretation that satisfies $(P \,\&\, Q)$, also satisfies $(P \lor Q)$. Hopefully it is obvious to you why this is. Only an interpretation that assigns 1 to both P and Q will satisfy the left, and any such interpretation will certainly satisfy the right as well. (You could draw the truth table to verify this if you want the practice.) So, as one might naturally expect, $(P \,\&\, Q) \models (P \lor Q)$.

Here is a less intuitive example. What should we make of this claim?

$$(P \,\&\, Q) \models (A \equiv \neg\neg A)$$

Notice that the sentence letters on the left-hand-side here are completely different letters from those on the right-hand-side. So there is a straightforward sense in which the two sides of the turnstile have *nothing to do with one another*. Nevertheless, this *is* a true entailment claim. This is because it satisfies the definition: every interpretation that satisfies the left (i.e., every interpretation that assigns '1' to both P and Q), also satisfies $(A \equiv \neg\neg A)$. This for the simple reason that every interpretation *whatsoever* satisfies $(A \equiv \neg\neg A)$; it is a tautology.

From this example we can see that a tautology will be entailed by anything whatsoever. If the right-hand-side of the entailment claim is a tautology, it doesn't matter what's on the left — you know it's going to be true.

So these are all true:

$$P \models (P \vee \neg P)$$
$$Q \models (P \vee \neg P)$$
$$(P \,\&\, \neg P) \models (P \vee \neg P)$$

4.5 Tautologies are entailed by the empty set

In the examples so far, we've been talking about one sentence entailing another. But we can also describe a set of several sentences as jointly entailing an SL sentence:

$$\Phi_1, \Phi_2, \Phi_3, \cdots \models \Psi$$

means that there is no truth value assignment for which all of the sentences in the set $\{\Phi_1, \Phi_2, \Phi_3, \cdots\}$ are true and Ψ is false. It will sometimes be convenient to use a variable that can stand in for any set of sentences; just as 'Φ' can stand for any sentence of SL, we can let '$\mathcal{X}$' stand in for any set of sentences in SL. (In the case of '$\Phi \models \Psi$' above, '$\mathcal{X}$' is the singleton set containing just Φ.)

In general, we may say that

$$\mathcal{X} \models \Psi$$

means that there's no interpretation satisfying every member of set $\mathcal{X}$ without also satisfying sentence Ψ.

We saw in the previous section that a tautology — an SL sentence that is true in every interpretation — is entailed by any sentence. The reasoning applies more generally: a tautology will be entailed by any set of sentences. This includes the *empty set*, which can be written '$\varnothing$'. In general '$\mathcal{X} \models \Phi$' says that every interpretation that satisfies everything in $\mathcal{X}$, satisfies Φ; in the special case where $\mathcal{X}$ is $\varnothing$, every interpretation trivially satisfies everything in $\mathcal{X}$. So '$\varnothing \models \Phi$' says that absolutely every interpretation satisfies Φ. That is, it says that Φ is a tautology.

By convention, we can leave the left side of an entailment claim blank, instead of writing in $\varnothing$. In other words,

$$\models \Phi$$

is shorthand for

$$\varnothing \models \Phi.$$

So these entailment claims are true, because in each case the sentence on the right is a tautology:

$$\models (P \lor \neg P)$$
$$\models (P \equiv P)$$
$$\models ((P \,\&\, \neg P) \supset (A \lor B))$$

And these entailment claims are not true, because each sentence on the right is not a tautology:

$$\models P$$
$$\models (P \,\&\, \neg P)$$
$$\models \neg (P \supset \neg P)$$

(If you need help seeing why a given sentence is or is not a tautology, draw out the truth table and check to see whether it has a 1 on every row. See §3.3.)

4.6 Inconsistent sentences entail absurdities

Consider this entailment claim:

$$(P \,\&\, \neg P) \models Q$$

This statement is true. It says that every interpretation satisfying $(P \,\&\, \neg P)$ also satisfies Q. But $(P \,\&\, \neg P)$ is a contradiction — no interpretations satisfy it. So trivially, 'every' interpretation that satisfies it satisfies Q.

This explanation had nothing to do with the specifics of the sentence Q. Exactly the same considerations would demonstrate that, for any sentence Ψ, $(P \,\&\, \neg P) \models \Psi$. And the same goes for any other contradictory sentence, or any set of sentences that are mutually inconsistent. These are all true:

$$(P \,\&\, \neg P) \models (\neg Q \supset R)$$
$$P, (\neg Q \,\&\, \neg P) \models (Q \equiv P)$$

$$(P \,\&\, \neg P) \models ((A_1 \vee A_2) \supset \neg(A_3 \equiv (A_4 \,\&\, \neg A_2)))$$

You can think of evaluating an entailment claim as like checking to see whether a rule is being violated or not. If there's a rule that says every student with a dog has to have a permit, you check each student with a dog, and make sure they have a permit. If you find someone without a dog, it doesn't matter whether they have a permit or not. (If you don't have a dog, you're not breaking the rule.) Or if you find someone with a permit, it doesn't matter whether they have a dog. (Nobody with a permit can be breaking the rule.) An entailment claim is like a rule that says every interpretation that satisfies the left-hand-side must also satisfy the right-hand-side. To verify it, you can ignore any interpretation that falsifies the left, or that satisfies the right.

In each of the cases above, you don't even need to examine the sentence at the right to confirm that the entailment claim is true, since you already know that no interpretations satisfy the sentences on the left. (This is analogous to knowing that no students have a dog — in this case you don't have to check to see whether anyone has a permit to confirm that the rule is being respected.) Any entailment claim with an unsatisfiable set of sentences on the left is true. Unsatisfiable sets of sentences entail *every* sentence.

We have a special notation for describing this situation. We write:

$$\mathcal{X} \models \bot$$

to indicate that the set $\mathcal{X}$ entails an *absurdity*; think of '$\bot$' as standing in for an arbitrary unsatisfiable SL wff. So you can think of '$\mathcal{X} \models \bot$' as saying that every interpretation that satisfies $\mathcal{X}$ satisfies the unsatisfiable. That's just another way of saying that *no* interpretation satisfies $\mathcal{X}$. (Compare: 'the only Klingons who hate honour are the Klingons who aren't Klingons at all.')

So these entailment claims are true, because the set of sentences on the left is not satisfiable:

$$(P \,\&\, \neg P) \models \bot$$
$$P, (\neg Q \,\&\, \neg P) \models \bot$$
$$(P \equiv \neg P) \models \bot$$

And these entailment claims are not true, because the sentences on the left *can* be satisfied:

$$(P \vee \neg P) \models \bot$$
$$P, \neg Q, (R \vee Q) \models \bot$$
$$\neg P, \neg Q, (P \supset \neg\neg Q) \models \bot$$

(If you need help seeing why a given set of sentences can be satisfied, draw out the truth table and check to see if any row assigns a 1 to each sentence. See §3.3.)

4.7 Defining concepts in terms of entailment

The double turnstile symbol allows us to give concise definitions for various concepts in SL:

A TAUTOLOGY IN SL is a sentence Φ such that $\varnothing \models \Phi$.

A CONTRADICTION IN SL is a sentence Φ such that $\Phi \models \bot$.

A sentence is CONTINGENT IN SL if and only if it is neither a tautology nor a contradiction.

An argument with premises $\mathcal{X}$ and conclusion Φ is VALID IN SL if and only if $\mathcal{X} \models \Phi$.

Two sentences Φ and Ψ are LOGICALLY EQUIVALENT IN SL if and only if both $\Phi \models \Psi$ and $\Psi \models \Phi$.

A set of sentences $\mathcal{X}$ is INCONSISTENT IN SL if and only if $\mathcal{X} \models \bot$. Otherwise $\mathcal{X}$ is CONSISTENT IN SL.

Practice Exercises

Part A Determine whether each entailment claim is true. You may construct a truth table to test it if you like, but these examples are simple enough so that you may be able to just think it through and get the right answer.

1. $Q \models (P \vee Q)$
2. $P, Q \models (P \vee P)$
3. $S \models (Q \supset Q)$
4. $(P \mathbin{\&} \neg P) \models (Q \mathbin{\&} \neg Q)$
5. $(P \vee \neg P) \models$
6. $\models (P \vee \neg P)$
7. $\models (P \mathbin{\&} \neg P)$
8. $(A \vee B) \supset \neg P \models (P \vee \neg P)$
9. $(P \equiv Q) \models ((P \mathbin{\&} Q) \vee \neg (P \vee Q))$

Chapter 5

SL Trees

So far we have learned one way to evaluate SL arguments for validity: an argument is valid just in case no interpretation satisfies the premises but falsifies the conclusion. We can check to see whether an argument is valid by constructing truth tables, representing all of the possible interpretations, and checking to see whether any of them do so. This method has two main advantages: one is that it can be done in a formal and formulaic way — it requires no particular rational insight, just a straightforward application of the rules of SL. Another is that, assuming you follow the rules correctly, it will always succeed in identifying whether the argument is valid or not.

The truth-table method also has a significant disadvantage, however: it rapidly becomes extremely cumbersome for evaluating arguments using more than two or three atomic sentence-letters. To evaluate an argument like this one, you'd need to consider four rows of a truth-table:

$$P \equiv Q$$
$$\neg Q$$
$$\neg Q \lor (P \supset \neg Q)$$
$$\therefore \neg P \supset Q$$

But change just two of the atoms to new letters, and the required table will grow exponentially. For this argument, you'd need sixteen rows.

$$P \equiv Q$$
$$\neg A$$
$$\neg Q \lor (A \supset \neg B)$$
$$\therefore \neg P \supset B$$

For this one, you'd need two hundred fifty-six!

$$A \equiv B$$
$$\neg B \supset (C \vee D)$$
$$E \supset \neg C$$
$$(\neg D \mathbin{\&} F) \vee G$$
$$\neg A \mathbin{\&} E$$
$$\therefore\ H \vee G$$

So it is useful to have alternate ways of proving entailments. In this chapter we will introduce a *proof system* for SL. The proofs we will construct in this chapter are called ANALYTIC TABLEAUX. Another name for them is TREES, because of the way they 'branch' out to represent possibilities. In Ch. 7 we will examine a different proof system.

5.1 Satisfiability and entailment

The method of trees is most directly a way to test for satisfiability of a set of SL sentences. Since validity is definable in terms of satisfiability, it also gives a way to test for validity.

Recall from §4.1 the definition of entailment in SL: $\mathcal{X} \models \Phi$ means that there is no interpretation that satisfies every sentence in $\mathcal{X}$ but falsifies Φ. (Remember, '$\mathcal{X}$' can stand for any set of SL sentences.) This in turn is equivalent to the claim that there is no interpretation that satisfies $\{\mathcal{X}, \neg\Phi\}$. Unsatisfiability can be represented with a double-turnstile with $\perp$ on the right-hand side.

> In general, $\mathcal{X} \models \Psi$ is equivalent to $\mathcal{X}, \neg\Psi \models \perp$.

So if you want to tell whether an argument in SL is valid, check to see whether the premises, along with the negation of the conclusion, are jointly satisfiable. If they are, the argument is invalid. If they're not, the argument is valid. This is the basic idea of the tree method. We'll write down the sentences we're attempting to satisfy, and then we'll work through their consequences, in order to see whether it's possible to complete the task.

Let's begin by working through an example that illustrates the general idea, then come back to give precise rules for the tree method.

5.2 An example: proving validity

Let's attempt to prove whether this is valid in SL:

$A \& B$
$\neg(C \vee D)$
$(\neg B \vee C) \vee E$
$\therefore E$

To evaluate this argument, we consider whether it's possible to satisfy the three premises along with the negation of the conclusion. We begin by writing down the sentences we're going to attempt to satisfy.

1. $\quad A \& B$
2. $\quad \neg(C \vee D)$
3. $\quad (\neg B \vee C) \vee E$
4. $\quad \neg E$

Notice that we're putting the *negation* of the conclusion here in line (4), since we're looking to see whether it's possible to satisfy the premises and *falsify* the conclusion. The sentences we write down at the beginning of the tree, we call the ROOT. Trees are designed to show whether the root is satisfiable, and if so, how.

With proof trees, the idea is to continue writing down that which our previous lines *already commit us to*. Consider the first line, $A \& B$. For this conjunction to be true, both conjuncts must be true. So any interpretation that satisfies the top four lines must also satisfy both A and B; the truth of those sentences is a commitment of what we already have. Each follow from what is already written down. We represent this by continuing the tree with those sentences:

1. $\quad A \& B$ ✓
2. $\quad \neg(C \vee D)$
3. $\quad (\neg B \vee C) \vee E$
4. $\quad \neg E$
   $\quad\quad |$
5. $\quad A$
6. $\quad B$

In addition to introducing lines (5) and (6), we also add a check mark on (1), to indicate that we've now considered it and represented its consequences. Think of the check mark as saying that we've extended the tree in a way that encodes the information from this line.

Now consider line (2). This is a negated disjunction. Disjunctions are true any time either disjunct is true, so this *negated* disjunction can only be true if *both* disjuncts are *false.* So we include new lines for $\neg C$ and $\neg D$, adding a check mark on line (2):

1.	$A \,\&\, B$ ✓
2.	$\neg(C \vee D)$ ✓
3.	$(\neg B \vee C) \vee E$
4.	$\neg E$
	$\mid$
5.	A
6.	B
	$\mid$
7.	$\neg C$
8.	$\neg D$

Line (3) is a disjunction. Unlike the previous two cases, it doesn't tell us what categorically *must* be the case; it says that one of two possibilities must be met. We represent this in our tree by *branching* into two different columns:

1.	$A \,\&\, B$ ✓
2.	$\neg(C \vee D)$ ✓
3.	$(\neg B \vee C) \vee E$ ✓
4.	$\neg E$
	$\mid$
5.	A
6.	B
	$\mid$
7.	$\neg C$
8.	$\neg D$
9.	$\neg B \vee C \qquad E$

Think of these two branches as representing the idea that the root implies that at least one of these ways of continuing the tree must be possible.

So now we have two branches to consider. Start by examining the right branch. It gives an atomic sentence, E. Notice, however, that line (4) was $\neg E$. So if we're looking for a way to satisfy (1)-(4), this right branch isn't a possible interpretation. It requires E to be true, and it also requires $\neg E$ to be true. If a branch contains any sentence and also contains its negation, we know that branch doesn't correspond to a possible interpretation. We'll consider this branch *closed*, and mark this with an '×'.

The left branch is another disjunction. It too branches out into its two disjuncts:

1.	$A \,\&\, B$ ✓	
2.	$\neg(C \lor D)$ ✓	
3.	$(\neg B \lor C) \lor E$ ✓	
4.	$\neg E$	

$$|$$

5.	A
6.	B

$$|$$

7.	$\neg C$
8.	$\neg D$

9. $\neg B \lor C$ ✓ E
                           ×

10. $\neg B$ C
        ×   ×
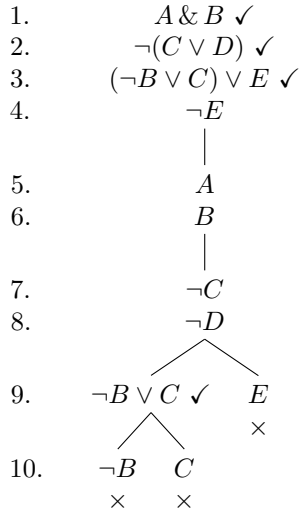
Both of these disjuncts also result in closed branches. The left branch at (10) is the negation of (6), and the right branch is the negation of (7). Now every branch in this tree is closed. This corresponds to the idea that every possible way there could be to satisfy (1)-(4) ended up requiring a contradiction. There is no interpretation that satisfies (1)-(4). In other words, the argument we began with was valid.

In the previous chapter we used the double turnstile, '$\models$', to represent entailment. In this chapter we will use a different but related symbol, the SINGLE TURNSTILE, which looks like this: '$\vdash$'. Think of this symbol as representing *provability*. So in proving that E follows from the premises above, we're showing that $\{A \,\&\, B, \neg(C \lor D), (\neg B \lor C) \lor E\} \vdash E$.

Unsurprisingly, provability and entailment will turn out to be closely connected; we'll consider the connection in detail in Chapter 6.

5.3 An example: proving invalidity

Let's work through another example to further illustrate the idea. In §5.4 we'll learn the formal rules for trees in SL. Consider this argument form:

$$(D \lor A) \,\&\, \neg N$$
$$N \lor \neg A$$
$$\therefore \; \neg N \,\&\, A$$

As before, we'll construct a tree that includes the premises and the negation of the conclusion at the top, and we'll attempt to find an interpretation satisfying those three lines. By convention, we write the claim we're attempting to prove at the top of the tree. We begin by processing line (1), which is a conjunction; its conjuncts are given in (4) and (5). Line (6) processes the disjunction in line (2), and the left branch closes.

$\{(D \lor A) \mathbin{\&} \neg N, N \lor \neg A\} \vdash \neg N \mathbin{\&} A$

1. $(D \lor A) \mathbin{\&} \neg N$ ✓
2. $N \lor \neg A$ ✓
3. $\neg(\neg N \mathbin{\&} A)$
                |
4. $D \lor A$
5. $\neg N$
              ⟨branch⟩
6. N $\neg A$
      ×

Line (3) is a negated conjunction. A conjunction is true if and only if both conjuncts are true, so it is false if at least one conjunct is false. So to resolve (3), line (7) branches into the negation of each conjunct, $\neg\neg N$ and $\neg A$. The former closes because it's the negation of line (5). The final sentence requiring resolution is the disjunction on line (4); it branches, and one branch closes.

$\{(D \lor A) \mathbin{\&} \neg N, N \lor \neg A\} \vdash \neg N \mathbin{\&} A$

1. $(D \lor A) \mathbin{\&} \neg N$ ✓
2. $N \lor \neg A$ ✓
3. $\neg(\neg N \mathbin{\&} A)$ ✓
                |
4. $D \lor A$ ✓
5. $\neg N$
              ⟨branch⟩
6. N $\neg A$
      ×
7. $\neg\neg N$ $\neg A$
                ×
8. D A
                        ↑   ×

The ↑ indicates that the open branch ending in D is a *completed* branch. (We'll precisely define 'completion' in §5.6.) What this means is that this branch

represents a way to satisfy all three sentences at the root of the tree. In other words, this argument form is *not* valid in SL. Furthermore, examining the open branch demonstrates an interpretation that satisfies the root. The branch includes three wffs that are either atoms or negated atoms: ¬*A*, *D*, and ¬*N*. So it suggests this interpretation:

$$I = \begin{cases} A = 0 \\ D = 1 \\ N = 0 \end{cases}$$

You can verify this result by evaluating the premises and the conclusion of the argument we began with on this interpretation. Since there is an interpretation that satisfies the premises and falsifies the conclusion, our tree system proves the argument invalid.

5.4 Resolution rules for SL trees

Hopefully the examples we've worked through have given you a decent intuitive sense of the tree method for SL. Now let's get more precise about it, by giving formal rules for trees. You should be able to recognize the following rules as a generalization of the steps of the proofs given above.

We begin with RESOLUTION rules. These rules identify ways that tree branches can be extended, given sentences that are already in that branch. The rules depend on the main connective of the sentence in question. (If the main connective is negation, then they also depend on the main connective of the negand.)

5.4.1 Conjunction

The rule for conjunction is that if you have a conjunction in a branch, you may make a linear extension of the branch that includes each conjunct, adding a check mark next to the conjunction. Using our Greek symbols once again as variables to stand in for any sentence of SL, any time you have this,

$$\Phi \,\&\, \Psi$$

you may extend each open branch of the tree to this:

$$\Phi \mathbin{\&} \Psi \ \checkmark$$
$$|$$
$$\Phi$$
$$\Psi$$

It is important to remember once again that Φ and Ψ here can stand for *any* sentences of SL, including complex ones.

If you have extended a tree branch using the conjunction rule, we say that you have *resolved* the conjunction. When sentences are resolved, they are marked with a check. In stating the resolution rules, we'll typically omit the check mark; here is the conjunction rule:

$$\Phi \mathbin{\&} \Psi$$
$$|$$
$$\Phi$$
$$\Psi$$

5.4.2 Negated conjunction

If you have a negated conjunction, you may resolve it by branching into the negation of each conjunct. This makes sense because there are two ways for a negated conjunction to be true — either conjunct can be false.

$$\neg(\Phi \mathbin{\&} \Psi)$$
$$\diagup \ \diagdown$$
$$\neg\Phi \qquad \neg\Psi$$

5.4.3 Disjunction

Disjunctions branch into each disjunct.

$$\Phi \vee \Psi$$
$$\Phi \qquad \Psi$$
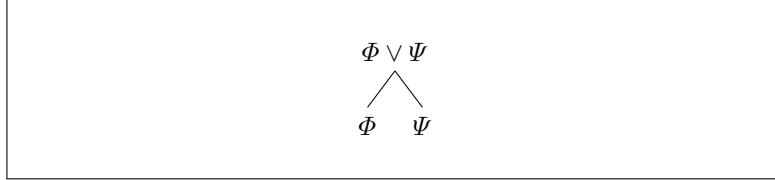
5.4.4 Negated disjunction

Since disjunctions are true any time either disjunct is true, they are only false if both disjuncts are false. So negated disjunctions are resolved with a linear development containing the negation of each disjunct.

$$\neg(\Phi \vee \Psi)$$
$$\neg\Phi$$
$$\neg\Psi$$
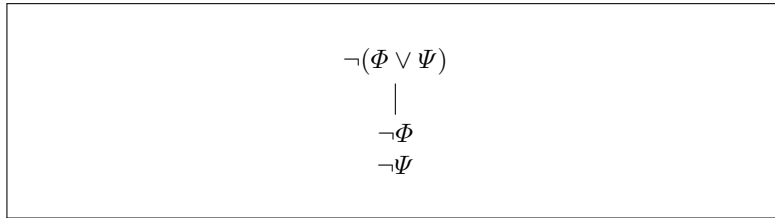
5.4.5 Conditional

Recall the characteristic truth table for the conditional in SL:

Φ	Ψ	$\Phi \supset \Psi$
1	1	1
1	0	0
0	1	1
0	0	1

Conditionals are true any time the antecedent is false, and also any time the consequent is true. We can represent this with a branching tree development, similar to a disjunction.

$$\Phi \supset \Psi$$
$$\diagup \diagdown$$
$$\neg\Phi \quad \Psi$$

5.4.6 Negated conditional

As in the case of disjunction, the negation of a conditional is a relatively strong claim — the only way for a conditional to be false is for its antecedent to be true *and* for its consequent to be false. We represent this with a linear development for negated conditionals:

$$\neg(\Phi \supset \Psi)$$
$$|$$
$$\Phi$$
$$\neg\Psi$$

5.4.7 Biconditional

Biconditionals require a slightly different structure. A biconditional says that two sentences have the same truth value: either both are true, or both are false. Since these represent two different ways a biconditional can be true, our tree rules will develop biconditionals into two new branches. But unlike in the case of our other rules, each new branch will contain two sentences. One branch will have both sides of the biconditional; the other will have both sides' *negations*:

$$\Phi \equiv \Psi$$
$$\diagup \diagdown$$
$$\Phi \quad \neg\Phi$$
$$\Psi \quad \neg\Psi$$

5.4.8 Negated biconditional

Negated biconditionals yield the same structure, but instead of new branches where each subsentence has the same truth value, they yield branches in which the subsentences have opposite values.

$$\neg(\varPhi \equiv \varPsi)$$

$$\varPhi \qquad \neg\varPhi$$
$$\neg\varPsi \qquad \varPsi$$
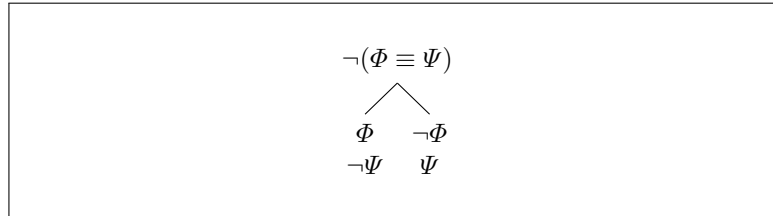
5.4.9 Double negation

There is one more resolution rule, for double-negated sentences. One resolves a double negation by developing a linear branch with the negand of the negand — i.e., the wff obtained from removing both negations from the left of the double-negation.

$$\neg\neg\varPhi$$
$$|$$
$$\varPhi$$
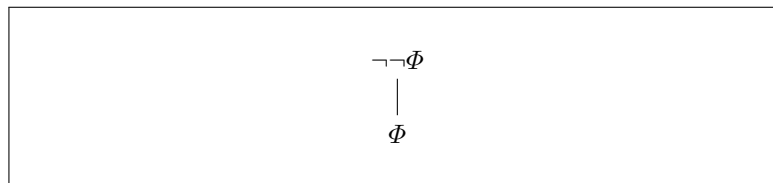
5.4.10 Resolution rules summary

These nine resolution rules describe, in purely formal terms, how to resolve most sentences of SL. The only exceptions are atomic sentences, and atomic sentences with a single negation sign in front of them. In our formal proof system, atoms and negated atoms have a special role to play; they are not resolved. In developing a tree proof, you may apply any of these rules to any unresolved sentence (other than an atom or negated atom) in the branch. Mark resolved sentences with a check.

Note that it is *not* a rule that sentences must be resolved in the order in which they appear in the tree; one can save sentences and come back later to resolve them. However, if you are resolving a sentence after the tree has already done

some branching, you must perform the resolution rule under *each* open branch
that includes that sentence. Here is a tree illustrating the issue.

```
1.              (D & ¬R) ∨ Q ✓
2.                   ¬Q ∨ R ✓

3.        D & ¬R ✓        Q            1 ∨

4.       ¬Q     R     ¬Q     R         2 ∨
          |      |      ×      ↑
          |      |     3, 4
5.        D      D                     3 &
6.       ¬R     ¬R
          ↑      ×
                4, 6
```
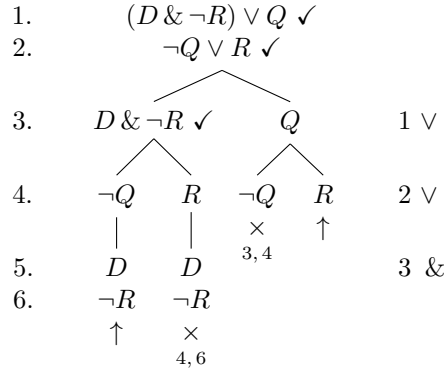
This tree has two disjunctions in the root, so it will involve branching multiple
times. We could start with either one, but this tree resolved line (1) first,
branching into the two disjuncts at line (3). The '1 ∨' to the right at that line is
the explanation for what is happening there: it indicates that the disjunction
rule was applied to line (1). When the second disjunction is processed at line
(4), it needs to be done on both branches, since both branches include line (2).
That's why our two branches split into four at line (4). One branch closes at
that point. At line (5), the conjunction at line (3) is processed. This needs to
be done on both branches that include that disjunction, but not the right-most
branch, which does not include it. Resolving a sentence affects everything below
it in the tree in its own descendent branches, but it doesn't affect branches that
are off to the side.

Besides the tree resolution rules, there are two more rules to our proof system.

5.5 Branch closure rules

In the tree examples above, we *closed* branches when they contained sentences
along with their negations. Here is our general rule for tree closure:

> If any branch contains some formula Φ, and also contains $\neg\Phi$, that
> branch is CLOSED. Any branch that is not closed is OPEN. We
> mark closed branches with the '×' symbol.
> A tree is CLOSED if and only if every branch in that tree is closed.
> A tree is OPEN if and only if it is not closed.

If every branch is closed, the method proves that the root is unsatisfiable.

Note that 'Φ' stands in for *any* sentence of SL; it is not part of the closure rule that one must have an atomic sentence and its negation. Consider for example this tree:

1. $(\neg\neg S \,\&\, T) \supset (\neg P \equiv (Q \vee R))$ ✓
2. $\neg\neg S \,\&\, T$
3. $\neg(\neg P \equiv (Q \vee R))$

4. $\neg(\neg\neg S \,\&\, T)$ $(\neg P \equiv (Q \vee R))$
            $\times$ $\times$
           $2, 4$ $3, 4$
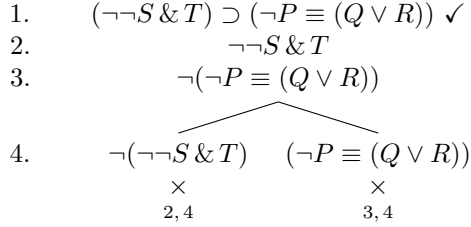
In this tree, we only resolved the first sentence, using the conditional rule, and the tree immediately closed. The line numbers under the closure symbols specify exactly why the branches close: the left branch developed into the negation of (2), and the right branch developed into the negation of (3). Notice that we *could* have developed lines (2) and (3), using the conjunction or negated biconditional rules, respectively, but this would have resulted in a more complicated tree that ultimately yielded the same result. Similarly, if we hadn't noticed that both branches close at line (4), we could have continued processing the tree, developing the left branch using the negated conjunction rule on (4), and developing the right branch using the biconditionals rule. But this too would have involved needless complication for no benefit. (Eventually, the tree would have closed with simpler sentences too.) You can save yourself a lot of work by noticing when branches are ready to mark as closed, and by thinking a bit strategically about which sentences to resolve first.

5.6 Branch completion rules

Here is the final piece of our formal proof system. In our examples above we have used '↑' to indicate that an open branch is complete. We need a formal characterization of branch completion.

In SL, a RESOLVABLE wff is a wff for which we have a resolution rule. That is, anything other than an atom or a negated atom.

> A branch is COMPLETE if and only if either it is closed, or every resolvable sentence in that branch has been resolved.
> A tree is complete if and only if every branch in that tree is complete.

If there is at least one open branch in a completed tree, the tree method indicates that that branch corresponds to an interpretation that satisfies the root.

We use the '⊢' symbol to indicate tree closure. '$\mathcal{X} \vdash \bot$' means that a tree with root $\mathcal{X}$ closes. We use '$\nvdash \bot$' to indicate that a completed tree remains open.

We will also define our single turnstile symbol so that $\mathcal{X} \vdash \varPhi$ is equivalent to $\mathcal{X}, \neg\varPhi \vdash \bot$. So the latter is a way of saying that the tree method shows that the argument from $\mathcal{X}$ to $\varPhi$ is valid.

This completes the introduction of the formal rules for the SL tree proof system. In the remainder of this chapter, we'll work through some more examples and provide some advice for working through trees more efficiently. In Chapter 6 we'll explain why the tree method works, and prove why it's a good one.

5.7 Resolution order

The resolution rules do not specify the order in which you should resolve sentences in a tree; you are free to resolve in any order you like. But some ways of processing trees are more efficient than others. Here is an example to illustrate. Suppose we want to consider whether $\{\neg(C \,\&\, A), D \equiv C, A \lor B, \neg B\} \vdash \bot$. So we put those sentences in the root of a tree. Here's what happens if we resolve the sentences in the order in which they're given:

1.	$\neg(C \,\&\, A)$ ✓	
2.	$D \equiv C$ ✓	
3.	$A \lor B$ ✓	
4.	$\neg B$	
5.	$\neg C \qquad\qquad \neg A$	$1 \,\neg\,\&$
6.	$D \quad \neg D \qquad D \quad \neg D$	$2 \equiv$
7.	$C \quad \neg C \qquad C \quad \neg C$	
8.	$A \quad B \quad A \quad B \quad A \quad B$	$3 \lor$
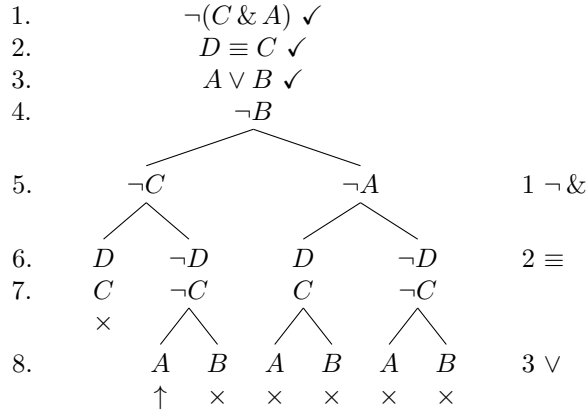
Again, the justifications for particular steps are added to the right of the tree for clarity. For example, '$1\neg\,\&$' indicates that line (5) was produced by performing the negated conjunction resolution rule on line (1). This tree remains open, offering this interpretation that satisfies the root:

$$I = \begin{cases} A = 1 \\ B = 0 \\ C = 0 \\ D = 0 \end{cases}$$

This tree reminds us again that if a sentence in a tree has multiple open branches below it, then resolving that sentence requires that the resolution be performed in *each* open branch below. That's why, for example, resolving line (3) at line (8) requires three different new branchings. (We do not resolve under the left-most column, because it is already closed.) So when there are more open branches, resolving sentences requires more work on the tree. Consequently, it is sometimes a good idea to think a bit strategically, and close off parts of the tree earlier, to save yourself some work.

The example above is a perfectly fine instance of a completed tree, and it does yield the correct answer. However, it's possible to get there with less work, by choosing to resolve sentences that will close off branches right away. Here is another way of developing a tree with the same root as in the previous example. This version begins by resolving line (3), because doing so will close off one of its new branches immediately. It then continues to resolve in the most efficient order:

1. $\neg(C \,\&\, A)$ ✓
2. $D \equiv C$ ✓
3. $A \lor B$ ✓
4. $\neg B$
5. $A \qquad B \qquad$ 3 $\lor$
   $\qquad\qquad \times$
6. $\neg C \quad \neg A \qquad$ 1 $\neg\,\&$
   $\qquad\quad \times$
7. $D \quad \neg D \qquad$ 2 $\equiv$
8. $C \quad \neg C$
   $\times \quad \uparrow$
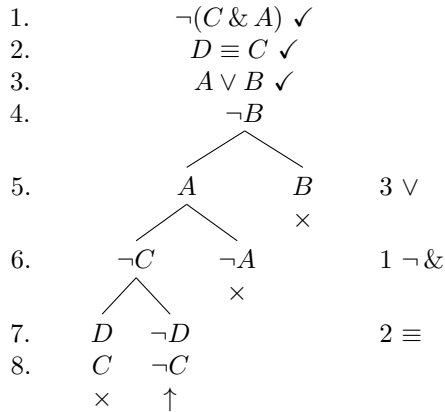
This tree gets us to the same result much more quickly, pointing to the same interpretation that satisfies the root. As a general rule of thumb, it's good advice to look a step ahead, and resolve sentences that won't lead to new open branches, before resolving ones that will. For just the same reason, it is usually more efficient to resolve those sentences that have linear rules before those that have branching rules.

5.8 Choosing the right root

Trees are used to indicate whether the root is satisfiable. We test arguments for validity with a tree by putting their premises along with the negation of their conclusions in the root; if the tree remains open, indicating that the set is satisfiable, that means it's possible to satisfy the premises while falsifying the conclusion, which means the argument is invalid. If the tree closes, that suggests the argument is valid.

But trees can be used to evaluate many other kinds of questions, aside from validity. Any question that can be converted into a question about satisfiability can be answered with a tree. For example, if you want to find out whether a set of sentences is consistent, put that set of sentences into the tree. A set of sentences is consistent if and only if there is an interpretation satisfying it, so you can use a tree to find out.

What if you want to find out whether a sentence is a tautology? A tautology is a sentence that is satisfied by *every* interpretation. To test this claim with a tree, we'll attempt to find an interpretation that falsifies it, by putting its *negation* in the root. If this tree remains open, it shows us how to satisfy the negation, which means that the sentence is *not* a tautology. If it closes, then the negation is unsatisfiable, which means the sentence *is* a tautology.

To find out whether a sentence is a contradiction, check and see whether it is possible to satisfy it, by putting it in the root of the tree. If the tree remains open, the sentence is satisfiable, which means it's not a contradiction. If the tree closes, then it is a contradiction.

To use a tree to answer a question, the first step is always to convert the question into a question about whether some sentence, or set of sentences, is satisfiable.

Practice Exercises

If you want additional practice, you can construct trees for any of the SL arguments and entailment claims in the exercises for the previous two chapters.

⋆ **Part A** To evaluate each of the following claims with a tree, (a) what would you put in the root of the tree?, and (b) if the tree closes, does that show that the claim is true or false?

1. $\{P, P \supset Q, Q \supset \neg P\} \vdash \bot$
2. $(P \supset Q) \equiv (Q \supset P)$ is a tautology.
3. The following argument is valid:

$$P \& Q$$
$$\neg R \supset \neg Q$$
$$\therefore \ P \& R$$

4. There is no interpretation that satisfies $A \vee B$, $B \supset C$, and $A \equiv C$ without also satisfying C.
5. $A \equiv \neg A$ is a contradiction.
6. Every interpretation satisfying P, $P \supset Q$, and $\neg Q$ also satisfies A.
7. There is at least one interpretation that satisfies $P \supset Q$, $\neg P \vee \neg Q$, and $Q \supset P$.

⋆ **Part B** Evaluate the argument given on p. 72 by constructing a tree. If it is invalid, give a model demonstrating it so.

⋆ **Part C** Evaluate each claim from Part A by constructing a tree. If applicable, give the interpretation that demonstrates the claim true or false.

Part D Recall the discussion of the Sheffer stroke in Chapter 3, page 60. That connective, if added to SL, would have this characteristic truth table:

Φ	Ψ	$\Phi|\Psi$
1	1	0
1	0	1
0	1	1
0	0	1

What would be appropriate tree resolution rules for the Sheffer stroke, or the negated Sheffer stroke?

Chapter 6

Soundness and Completeness for SL Trees

In Chapter 5 we introduced a proof system for SL. Trees provide a method for answering the question of whether a given set of SL sentences is jointly satisfiable. Our focus last chapter was on *using* trees to answer those questions; this chapter we turn to the study of the proof system itself. This chapter engages in a project of *metalogic*. In particular, we aim to first precisify, then answer, this question: is the tree method a good method?

It might not be obvious at first that there is even a genuine question here. The tree method is a formal method, with precisely defined rules. You might be tempted to think that, *by definition*, following the rules makes it a good method. There is one sense in which that is right — the rules laid out last chapter *tell you* to follow the rules laid out last chapter, so following the rules is doing what the system tells you to do. But there is also a deeper question to be asked. The rules were not selected at random. They were designed to do something in particular: namely, to tell whether a set of sentences is satisfiable. This isn't a question about trees. (Note that we were considering this question back in Chapter 4, before we had even introduced the idea of trees.) So there is a question to be asked about whether the tree method actually does what it's supposed to do.

Recall the distinction between our two turnstiles: we use '$\mathcal{X} \models \bot$' to mean that no interpretation satisfies $\mathcal{X}$; '$\mathcal{X} \vdash \bot$' means that a tree with root $\mathcal{X}$ closes. We interpret the latter as our proof system *saying* that nothing satisfies the root. Our question now is, can our proof system be trusted? Is it reliable? The main project of this chapter is to prove in a rigorous way that it can, and is.

6.1 Informal proof

In Chapter 5 we learned a formal proof system. This chapter, we will prove important results *about* that system. It is important to emphasize, however, that the formal proof system isn't the only way to 'prove' things. In particular, the tree method is *not* an appropriate methodology for the task of this chapter. When we say we wish to prove that the tree method is good, we don't mean that we will put the negation of that claim — i.e., that the tree method is not good — in the root of a tree. That would get us nowhere.

Instead, we will be engaging in an *informal* proof *about* the formal system. An informal proof needn't be any less conclusive or compelling than a formal proof is, but evaluating it makes use of our general ability to recognize what follows from what, rather than working through a list of syntactically-defined rules.

6.2 Soundness

If a tree with root $\mathcal{X}$ closes, we interpret that as the system telling us that $\mathcal{X}$ is unsatisfiable. If our system is a good one, then a tree will never mislead us in this respect. Tree closure should guarantee unsatisfiability. We call this property SOUNDNESS. The soundness of our SL tree system is the first important metalogical theorem we will prove in this chapter. (A 'metalogical' theorem is a theorem *about* logic.)

> SOUNDNESS: If a tree closes, that guarantees that its root is unsatisfiable. In other words:
>
> $$\mathcal{X} \vdash \bot \Rightarrow \mathcal{X} \models \bot$$

Here is a way to illustrate that soundness is a substantive result, and to clarify what it is we're trying to prove. Recall the resolution rule for disjunction (see p. 79):

$$\Phi \lor \Psi$$
$$\diagup \diagdown$$
$$\Phi \quad \Psi$$

Let's suppose for the purpose of argument that we had a different disjunction rule instead of this one. Suppose, for example, that our rule for disjunction had been this:

$$\begin{array}{c} \Phi \vee \Psi \\ | \\ \Phi \\ \Psi \end{array}$$

If this had been our disjunction rule, and all the other rules remained the same, our tree system would have been unsound. It would have been possible for satisfiable roots to result in closed trees. That is to say, there would have been possible sets of sentences $\mathcal{X}$ such that $\mathcal{X} \vdash \bot$, even though $\mathcal{X} \not\models \bot$. Consider for example these sentences:

> P
> $\neg P \vee Q$

These sentences are obviously jointly satisfiable, by an interpretation that assigns 1 to P and 1 to Q. But if we used the tree system with the alternate disjunction rule above, the tree would close:

1. P
2. $\neg P \vee Q$ ✓
              |
3. $\neg P$ 2 alt.$\vee$
4. Q
             $\times$
            $1, 3$

This would be a counterexample to the soundness of the SL tree system. This explains what's wrong with the alternate disjunction rule — it would allow trees to 'prove' that a root is unsatisfiable, even if it really is satisfiable. In considering the soundness of our system, we are investigating whether our actual proof system is defective in the same way this hypothetical modification of the system would have been. We will prove that it is not.

6.3 Recursive proofs

Our proof of soundness will be a *recursive* proof. A recursive proof is a proof that proceeds stepwise: one first demonstrates that the claim to be proven holds for some simple case, and then shows that, *if* it holds for some case, then it also holds for some other, slightly more complicated case. More ways of complicating cases may also be discussed, each along with the assurance that if the claim holds for a simpler case, then it will also hold for each more complex one. Finally, if one can demonstrate that the ways of complicating cases considered are *exhaustive* —

that is, if these represent the only possible cases — then this has been shown for every possible case. You may be familiar with recursive proofs already in the context of *mathematical induction.* (This is a topic that comes up in many high school algebra classes.)

Here is an example illustrating proof by induction. Suppose that Sir Roderic Murgatroyd has been cursed. The curse is subject to the following rules:

1. The only way for someone to escape the curse is to transfer it to someone else.

2. There are only three ways to transfer the curse to someone else:
   (a) One may transfer it to one's parent.
   (b) One may transfer it to one's child.
   (c) One may transfer it to one's sibling.

Given these rules, it's not difficult to see that the Murgatroyd curse will never leave the family. We know that Sir Roderic has the curse. He could transfer it to a parent, a child, or a sibling, but none of those actions would remove the curse from the family, since one's parents, one's children, and one's siblings are all family members. And any of *those* people, if *they* had the curse, can only transfer it to someone else within the family. No curse transfer can get the curse outside the family. So someone in the family will remain cursed forever.

Slightly more precisely: we're attempting to prove that the curse will always be in the family. Roderic is in the family. And, for any person, if they are in the family, then they cannot get rid of the curse without transferring it to another member of the family. The proof is perfectly general; it applies to Roderic's great-great-great-grandchildren just as well as it applies to Sir Roderic himself. This is a simple example of a recursive proof. The proof of the soundness of the SL tree method is more complex, but it has the same basic structure.

6.4 Proving soundness

Soundness is the claim that any time a tree closes, the root must be unsatisfiable. This is equivalent to the claim that any time the root *is* satisfiable, the tree *won't* close. (Compare: if every new citizen swears loyalty to the Queen, then everyone who *doesn't* swear loyalty to the Queen must *not* be a new citizen.) So to prove soundness, we can assume that the root is satisfiable, and show that it follows that the tree doesn't close.

Suppose, then, we have some satisfiable set of sentences $\mathcal{X}$ in the root of a tree. If the root is satisfiable, then there is some interpretation that satisfies it.

(Recall that an interpretation in SL is an assignment of truth values to atomic sentences.) Call this interpretation $\mathcal{I}$. We will begin by proving that, if our tree follows the rules given in Chapter 5, then $\mathcal{I}$ doesn't just satisfy the root — it satisfies every sentence in some branch of the completed tree. Once we establish that claim, it's only a short step to demonstrate that this branch doesn't close. Branches only close when they contain some formula and its negation. But no interpretation can satisfy a formula and its negation; so if $\mathcal{I}$ satisfies every formula in the branch, that means that branch must not contain any formula along with its negation. So the tree will remain open.

This is the broad structure of our proof. The key step is in proving that $\mathcal{I}$ has the property mentioned above — that it satisfies every sentence in an open branch. We will prove this recursively.

6.4.1 Root

Start with the root, $\mathcal{X}$. This is trivial. We are *assuming* that our tree begins with a satisfiable root, because we are trying to prove what follows from that assumption. (Namely, that the tree won't close.) $\mathcal{I}$ is just our name for one of the interpretations we are assuming must exist. So $\mathcal{I}$ satisfies everything in the root. This is a reasonable thing to assume, when proving soundness, because soundness only tells us what happens when the root is satisfiable: namely, that the tree won't close. Soundness doesn't say anything about what happens if the root is unsatisfiable.

We want our proof to be perfectly general, so we don't want to make any particular assumptions about what the tree does beyond the root. But, given the resolution rules outlined in §5.4, there are only nine possible ways the tree might develop at each step. (Compare the three possible ways the curse might move in the Murgatroyd example.) We will prove, for each of these nine resolution rules, the following: if $\mathcal{I}$ satisfies all the sentences in the branch *above*, then $\mathcal{I}$ also satisfies at least one branch of what comes *below*. In other words, we'll prove, for each inference rule, that that rule cannot take you from a satisfiable branch to a tree with no satisfiable branches.

6.4.2 Conjunction

Suppose that a tree develops via the conjunction rule:

$$\Phi \,\&\, \Psi$$
$$|$$
$$\Phi$$
$$\Psi$$

We assume that $\mathcal{I}$ satisfies the branch above the development. So in particular, $\mathcal{I}$ must satisfy $\Phi \,\&\, \Psi$. We may write this as

$$\mathcal{I}(\Phi \,\&\, \Psi) = 1$$

We know from the definition of truth in SL that any interpretation that assigns 1 to a conjunction must assign 1 to each conjunct. (See page 65.) So:

$$\mathcal{I}(\Phi) = 1$$

$$\mathcal{I}(\Psi) = 1$$

What we've just shown is that, if $\mathcal{I}$ satisfies the branch above this development, then it also satisfies everything in the new development. The conjunction rule will never take us from a satisfiable branch to an unsatisfiable one. We need to prove that *every* possible way of developing the tree is like that.

6.4.3 Negated conjunction

Negated conjunctions develop in our system with a branching rule:

$$\neg(\Phi \,\&\, \Psi)$$

$$\neg\Phi \qquad \neg\Psi$$

Once again, we are assuming for the purpose of argument that our interpretation $\mathcal{I}$ satisfies everything up until this development. So $\mathcal{I}(\neg(\Phi \,\&\, \Psi)) = 1$. Since $\mathcal{I}$ satisfies that negation, $\mathcal{I}(\Phi \,\&\, \Psi) = 0$. Given our definition of truth in SL, any interpretation that assigns 0 to this conjunction must assign 0 to at least one of its conjuncts. So *either* $\mathcal{I}(\Phi) = 0$ *or* $\mathcal{I}(\Psi) = 0$. (It might assign 0 to both, but what we know for sure is that it assigns 0 to at least one.) If $\mathcal{I}(\Phi) = 0$, then $\mathcal{I}(\neg\Phi) = 1$, and so the new left branch is satisfied. If $\mathcal{I}(\Psi) = 0$, then $\mathcal{I}(\neg\Psi) = 1$, and so the new right branch is satisfied. Since (at least) one of these must be the case, we know that $\mathcal{I}$ satisfies at least one branch of our extended tree, assuming it satisfied that which came before the extension. So the negated conjunction rule will never take us from a satisfiable branch to an unsatisfiable one.

6.4.4 Disjunction

Disjunctions branch according to this rule:

$$\Phi \vee \Psi$$

$$\Phi \qquad \Psi$$

Assume $\mathcal{I}(\Phi \vee \Psi) = 1$. Then either $\mathcal{I}(\Phi) = 1$ or $\mathcal{I}(\Psi) = 1$. If $\mathcal{I}(\Phi) = 1$, then $\mathcal{I}$ satisfies the left branch. If $\mathcal{I}(\Psi) = 1$, then $\mathcal{I}$ satisfies the right branch. So, assuming that $\mathcal{I}$ satisfies the sentences above this resolution rule, it must satisfy at least one branch below it. So the disjunction rule will never take us from a satisfiable branch to an unsatisfiable one.

Hopefully the pattern is becoming clear by now. We've proven, for three of our nine rules, that they cannot take us from a satisfiable branch to an unsatisfiable one. Six resolution rules remain to be considered.

Consider again the variant disjunction rule hypothesized above:

$$\Phi \vee \Psi$$

$$\Phi$$
$$\Psi$$

If we attempted to go through the same reasoning we've been going through, we'd fail. Assume $\mathcal{I}(\Phi \vee \Psi) = 1$. By the definition of truth in SL, we know that $\mathcal{I}$ assigns 1 to at least one of Φ and Ψ, but there is no guarantee that it will satisfy *both*. So we have no assurance that, if one follows this rule, $\mathcal{I}$ will satisfy the development of the tree, even if we assume it satisfies the sentences above. We cannot prove that the use of this rule will never lead to an inappropriate tree closure.

6.4.5 Negated Disjunction

Here is the rule for negated disjunctions:

$$\neg(\Phi \vee \Psi)$$

$$\neg\Phi$$
$$\neg\Psi$$

Suppose that $\mathcal{I}(\neg(\Phi \vee \Psi)) = 1$. Given the definition of negation, this means that $\mathcal{I}(\Phi \vee \Psi) = 0$. This in turn means, given the definition of disjunction, that $\mathcal{I}$ must assign 0 to both Φ and Ψ. And so of course, given the definition of negation again, we know that $\mathcal{I}$ assigns 1 to $\neg\Phi$, and also assigns 1 to $\neg\Psi$. Therefore, if we began with a satisfiable tree branch, invoking this rule, like the other good rules we've considered, will preserve satisfiability; the negated disjunction rule will never take one from a satisfiable branch to an unsatisfiable one.

6.4.6 Conditional

The conditional rule is:

$$\Phi \supset \Psi$$
$$\diagdown$$
$$\neg\Phi \qquad \Psi$$

As before, assume that the material above the branch is satisfiable; so some interpretation $\mathcal{I}$ satisfies it. Any interpretation that satisfies a conditional must assign 0 to the antecedent, or 1 to the consequent (or both). If $\mathcal{I}$ assigns 0 to the antecedent, then it satisfies the left development of the tree. If $\mathcal{I}$ assigns 1 to the consequent, then it satisfies the right development of the tree. So, given that it satisfies the conditional, $\mathcal{I}$ is guaranteed to satisfy at least one branch of the tree as developed by the conditional rule.

6.4.7 Negated Conditional

The rule given in Chapter 5 for negated conditionals was this:

$$\neg(\Phi \supset \Psi)$$
$$\mid$$
$$\Phi$$
$$\neg\Psi$$

I hope the procedure is feeling a bit tedious by now. As before, we assume that the negated conditional is satisfiable, and prove that the tree as developed by this rule will remain satisfiable. Since we're assuming that $\neg(\Phi \supset \Psi)$ is satisfiable, it follows that some interpretation $\mathcal{I}$ satisfies it. But $\mathcal{I}(\neg(\Phi \supset \Psi))$ only if $\mathcal{I}(\Phi) = 1$ and $\mathcal{I}(\Psi) = 0$. So $\mathcal{I}$ will satisfy Φ and $\neg\Psi$. That is to say, it will satisfy the continuation of the branch given this rule. The negated conditional rule can never take one from a satisfiable root to an unsatisfiable tree.

We have three more rules to consider.

6.4.8 Biconditional

Here is the biconditional rule:

$$
\begin{array}{c}
\Phi \equiv \Psi \\[2pt]
\diagup\;\diagdown \\[2pt]
\begin{array}{cc}
\Phi & \neg\Phi \\
\Psi & \neg\Psi
\end{array}
\end{array}
$$

Assuming that $\mathcal{I}$ satisfies $\Phi{\equiv}\Psi$, it must either assign 1 to both Φ and Ψ, or it must assign 0 to both Φ and Ψ. If the former, $\mathcal{I}$ will satisfy the left branch of the new development from this rule. If the latter, it will satisfy the right branch, since any interpretation that assigns 0 to a sentence must assign 1 to its negation. So this rule too can never take one from a satisfiable root to an unsatisfiable tree.

6.4.9 Negated biconditional

$$
\begin{array}{c}
\neg(\Phi \equiv \Psi) \\[2pt]
\diagup\;\diagdown \\[2pt]
\begin{array}{cc}
\Phi & \neg\Phi \\
\neg\Psi & \Psi
\end{array}
\end{array}
$$

The reasoning is much as before. If our interpretation satisfies the negated biconditional, then it must assign opposite values to each side; i.e., either $\mathcal{I}(\Phi) = 1$ and $\mathcal{I}(\Psi) = 0$, or $\mathcal{I}(\Phi) = 0$ and $\mathcal{I}(\Psi) = 1$. If the former, $\mathcal{I}$ satisfies the left branch; if the latter, $\mathcal{I}$ satisfies the right branch. So if the negated biconditional is satisfiable, this rule will never result in an unsatisfiable tree.

6.4.10 Double negation

Here is our final tree resolution rule:

$$
\begin{array}{c}
\neg\neg\Phi \\
\mid \\
\Phi
\end{array}
$$

One last time, we assume that we begin with something satisfiable; so we allow that some interpretation $\mathcal{I}$ assigns 1 to $\neg\neg\Phi$. If it assigns 1 to this negation, then it must assign 0 to its negand, $\neg\Phi$. That is to say, $\mathcal{I}(\neg\Phi) = 0$. And since it assigns 0 to *this* negation, it must assign 1 to *its* negand: $\mathcal{I}(\Phi) = 1$. But this just is the new branch development. So if we began with something satisfiable, this rule will result in something satisfiable.

6.4.11 Taking stock

We've shown, for the nine resolution rules in our tree system, that they each have the following important feature: if you begin with a satisfiable set of sentences, applying the rule will always result in at least one continuation of the tree that is also satisfiable. And since these nine rules are the only ways one can develop a tree, we've proven that there is no possible way, consistent with the tree rules, for a tree with a satisfiable root to develop into a tree with no satisfiable branches.

Branches can only be closed if they contain a sentence and its negation, which a satisfiable branch will never have. So, assuming we started with a satisfiable root, the rules will never result in a tree with all branches closed. Satisfiable roots will always result in open trees. The tree method will never erroneously "prove" that a root is unsatisfiable. Equivalently, tree closure guarantees unsatisfiability of the root. The tree method is sound.

> SOUNDNESS: If a tree closes, that guarantees that its root is unsatisfiable. In other words:
>
> $$\mathcal{X} \vdash \bot \Rightarrow \mathcal{X} \models \bot$$

6.5 Completeness

Soundness is the first of two important metalogical theorems considered in this chapter. The second is COMPLETENESS. One can think of soundness as a guarantee against a system proving *too much*; in proving soundness, we were assuring ourselves that a tree would close *only if* the root was unsatisfiable. Completeness, as the name suggests, concerns whether our system proves *enough*. We want our system to be sure to close, if the root is unsatisfiable. Remember, we take open branches in completed trees as an indication that the root is satisfiable. Completeness is about ensuring that this is a warranted conclusion.

> COMPLETENESS: If a root is unsatisfiable, that guarantees that the tree will close. In other words:
>
> $$\mathcal{X} \models \bot \Rightarrow \mathcal{X} \vdash \bot$$

Consider the unsatisfiable set of sentences, $\{\neg Q, P \,\&\, Q\}$. Given our conjunction rule, a tree with this root will close:

1.　　　　$\neg Q$
2.　　　$P \,\&\, Q$ ✓
　　　　　　│
3.　　　　　P　　　　2 &
4.　　　　　Q
　　　　　　×
　　　　　$1, 4$

In proving completeness, we wish to demonstrate that this will always be the case: *whenever* we begin with an unsatisfiable root, the entire tree will eventually close. Notice that if we had a different conjunction rule that called for a branching development instead of a linear one, completeness would fail. Suppose we had this rule:

$$\Phi \,\&\, \Psi$$
$$\diagup\diagdown$$
$$\Phi \quad \Psi$$

Using this rule, the tree with root $\{\neg Q, P \,\&\, Q\}$ would remain open:

1.　　　　$\neg Q$
2.　　　$P \,\&\, Q$ ✓
　　　　　$\diagup\diagdown$
3.　　　P　　Q　　　2 alt. &
　　　　↑　　×
　　　　　　$1, 3$

The right branch closes, but this tree has a left branch that remains open, even though the root is unsatisfiable. So if we modified our proof system by using this rule instead of the linear rule for conjunction, we would have a system that fails completeness. We wish to prove that, given the actual rules, our system is complete.

6.6 Proving completeness

Completeness is the claim that any time a set of sentences is unsatisfiable, a completed tree with that set as its root will close. This is equivalent to the claim that if a completed tree has a branch that remains open, then the root is satisfiable. To prove completeness, we will assume that a completed tree has a branch that remains open, and prove that, on this assumption, the root is satisfiable. In fact, we will prove something stronger than that: we will prove that *every* formula in a completed open branch is satisfiable, by demonstrating a recipe for constructing an interpretation that satisfies it. Since the root is part of every branch of the tree, this will suffice for proving completeness.

As in the case of our soundness proof, we will be giving an *informal* proof *about* our formal system.

Here is the broad shape of the proof. Suppose that a completed tree has at least one open branch. Then we can construct an interpretation, $\mathcal{I}$, based on that branch, as follows: if any atomic sentence Φ is in the branch, then $\mathcal{I}(\Phi) = 1$. If any negated atomic sentence $\neg\Psi$ is in the branch, then $\mathcal{I}(\Psi) = 0$. Let these assignments exhaust $\mathcal{I}$. We can guarantee that there will be a coherent interpretation like this. This recipe for constructing interpretations will fail only if some atomic sentence *and* its negation are *both* in the branch. But if a sentence and a negation are both in the branch, then that branch will close; by hypothesis, we're considering a completed branch that remains open. So we know it contains no explicit contradictions of this kind.

Now we want to prove that $\mathcal{I}$ satisfies every formula in the branch (including the root). We know it satisfies every atomic formula and every negated atomic formula in the branch, given the way it was constructed. We'll now show that it must satisfy every other formula too. We'll exploit the recursive rules for SL grammaticality: there are only a certain number of ways that sentences can be created from simpler sentences. We'll show, for every sentence form, that if $\mathcal{I}$ satisfies a branch downstream of a sentence of that form in a completed branch, then it satisfies that sentence too.

We begin with conjunction.

6.6.1 Conjunction

Suppose our completed, open branch contains a conjunction of the form $\Phi \,\&\, \Psi$. Since it is a *completed* branch, this means that the conjunction resolution rule must have been applied to this conjunction. (Remember, a branch isn't completed until every complex formula has a check mark next to it.) So, given the conjunction rule,

$$\Phi \,\&\, \Psi$$
$$|$$
$$\Phi$$
$$\Psi$$

we know that the branch must also contain Φ and Ψ. If we assume that $\mathcal{I}$ satisfies both these sentences, then we know from the definition of the truth of a conjunction in SL that $\mathcal{I}$ satisfies $(\Phi \,\&\, \Psi)$ too. In other words, there's no way to satisfy the simpler sentences that come after this resolution rule, without also satisfying the conjunction.

6.6.2 Negated conjunction

Suppose a negated conjunction appears in the open branch. Since the branch is complete, you know that the negated conjunction rule has been applied:

$$\neg(\Phi \,\&\, \Psi)$$

$$\neg\Phi \qquad \neg\Psi$$

We are assuming that the negated conjunction is in an open branch. This is consistent with either one of the branches below closing, but they cannot both close. If they did, the negated conjunction would not be in an open branch. So we know that at least one branch is open. So either $\mathcal{I}(\Phi) = 0$ (if the left branch is our open branch) or $\mathcal{I}(\Psi) = 0$ (if the right branch is the open branch). Since at least one of these sentences is assigned 0, their conjunction must also be assigned 0, which means the negated conjunction we're considering is assigned 1. So once again, if the material in at least one branch below the resolution rule is satisfied, then the negated conjunction is satisfied too.

6.6.3 Disjunction

Disjunctions are very similar to negated conjunctions. Since the tree is complete, any disjunction $\Phi \lor \Psi$ has a branch below it containing Φ, and one containing Ψ. Whichever of these disjuncts is in the open branch, $\mathcal{I}$ satisfies that disjunct, and so satisfies the disjunction too.

6.6.4 Negated disjunction

Negated disjunctions are similar to conjunctions. If a negated disjunction is in an open branch, then the negation of each disjunct is also in that branch. So, suppose that $\mathcal{I}$ assigns 0 to each disjunct. Then it also assigns 0 to their disjunction. So once again, if the material below the negated disjunction is satisfied, then so is the negated disjunction itself.

6.6.5 Conditional

If a conditional is in a completed open branch, then it has been resolved by this branching rule:

$$\Phi \supset \Psi$$
$$\diagdown\diagup$$
$$\neg\Phi \qquad \Psi$$
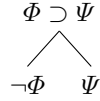
If the left development is the open branch, then we suppose that $\mathcal{I}(\Phi) = 0$, which means that $\mathcal{I}(\Phi \supset \Psi) = 1$. If this right development is the open branch, then we suppose that $\mathcal{I}(\Psi) = 1$, which *also* means that $\mathcal{I}(\Phi \supset \Psi) = 1$. So if the material below in at least one branch is satisfied, then the conditional is satisfied too.

6.6.6 Negated conditional

If a negated conditional $\neg(\Phi \supset \Psi)$ is in the open branch, then so too are Φ and $\neg\Psi$. So $\mathcal{I}(\Phi) = 1$ and $\mathcal{I}(\Psi) = 0$. So $\mathcal{I}$ falsifies the conditional, satisfying the negated conditional.

There are three more kinds of sentences that exist in SL.

6.6.7 Biconditional

Suppose a biconditional is in an open branch. If the branch is completed, then this rule has been performed:

$$\Phi \equiv \Psi$$
$$\diagup\diagdown$$
$$\Phi \qquad \neg\Phi$$
$$\Psi \qquad \neg\Psi$$
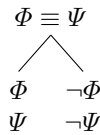
One of these developments is the open branch. If it's the left branch, then, supposing that $\mathcal{I}$ assigns 1 to both Φ and Ψ, $\mathcal{I}$ must also assign 1 to the biconditional $\Phi \equiv \Psi$. If it's the right branch, then, supposing that $\mathcal{I}$ assigns 0 to both Φ and Ψ, this also means that $\mathcal{I}$ must also assign 1 to the biconditional $\Phi \equiv \Psi$. So whichever branch is satisfied by $\mathcal{I}$, the biconditional is also satisfied.

6.6.8 Negated biconditional

Exactly the same reasoning as above applies to negated biconditionals, except this time, the branches each assign *opposite* truth values to Φ and Ψ. So for our interpretation to satisfy either branch, it must falsify the biconditional, thus satisfying the negated biconditional.

6.6.9 Double negation

Finally, suppose there is a double-negated sentence in our completed open branch. Then this rule has been performed:

$$\neg\neg\Phi$$
$$|$$
$$\Phi$$

If $\mathcal{I}(\Phi) = 1$, then, given the definition of truth in SL, $\mathcal{I}(\neg\Phi) = 0$, and $\mathcal{I}(\neg\neg\Phi) = 1$. So once again, if our interpretation satisfies what comes below, then it satisfies the double-negation above.

6.6.10 Summarizing the completeness proof

What we've just shown is that, for any sentence of SL, if it has one of the nine structures just canvassed — if it's a conjunction, a negated conjunction, a disjunction, etc. — then, if it is in a completed open branch where the sentences below it are satisfied by $\mathcal{I}$, then it too is satisfied by $\mathcal{I}$. Given the way that $\mathcal{I}$ was selected, we know that $\mathcal{I}$ must satisfy every atomic sentence, and every negated atomic sentence, in the open branch. And since the nine structures considered are the only ways to develop more complex sentences, this implies that *every* SL sentence in the open branch is satisfied by $\mathcal{I}$. This includes the root. Since interpretation $\mathcal{I}$ satisfies the root, this of course means that the root is satisfiable. That is to say, if a completed branch remains open, this guarantees that the root is satisfiable. Equivalently, if the root is unsatisfiable, a completed tree is guaranteed to close. Completeness is proven.

6.7 Testing alternate rules

We can use the reasoning involved in the soundness and completeness proofs above to consider various alternative tree rules. We saw one example of this on p. 89 above, when we observed that an alternate, linear rule for disjunctions would result in an unsound tree system. Here again was the rule we considered:

$$\Phi \lor \Psi$$
$$|$$
$$\Phi$$
$$\Psi$$

Note, however, that if we think through the *completeness* reasoning, we'll find that the completeness proof would still hold. If we assume that some interpretation $\mathcal{I}$ satisfies both Φ and Ψ, we can be assured that it also satisfies the disjunction $\Phi \lor \Psi$. So changing the disjunction rule to this linear one would *not* interfere with the completeness of our tree system. Our trees would still close any time they began with unsatisfiable roots. But as we saw on p. 94, the soundness proof would fail, which is why a system with this rule could start in a satisfiable root, and result in a closed tree.

When either the soundness or the completeness proof fails, you know you are working with an inappropriate rule. To conclusively demonstrate this, you can provide a counterexample to the failed metalogical theorem. A counterexample to soundness would be a tree with a satisfiable root that closes. A counterexample to completeness would be a completed tree, with an unsatisfiable root, that remains open. The rule above violates soundness, so using that rule we can construct a tree with a satisfiable root, that closes. Constructing the right counterexample takes a bit of thought. The rule puts both disjuncts into a single branch below, and we want it to close, despite having a satisfiable root. So adding the negation of just one of the disjuncts to the root will close the tree, without making the root unsatisfiable:

$$A \lor B \checkmark$$
$$\neg A$$
$$|$$
$$A$$
$$B$$
$$\times$$

This tree is a counterexample to soundness, using the hypothetical rule mentioned

above. Note that a counterexample is a tree that uses SL sentences, not the Greek letters Φ and Ψ that we use in the statements of the rules.

Let's work through one more example. Suppose we changed the conjunction rule to this one:

$$\Phi \,\&\, \Psi$$
$$\diagdown\!\!\!\diagup$$
$$\Phi \qquad \Psi$$
$$\Psi$$
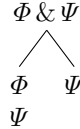
Would our system still be sound? To answer this, we assume that the conjunction $\Phi \,\&\, \Psi$ is satisfiable, and ask whether this guarantees that at least one branch below is also satisfiable. It does. (In fact, both branches are guaranteed to be satisfiable.) So the system will still be sound.

Would the system still be complete? To answer this, we ask whether each branch is such that, if we assume that an interpretation satisfies the developments below, it is guaranteed to satisfy the conjunction above. Begin with the left branch. If some interpretation satisfies both Φ and Ψ, then it will certainly satisfy the conjunction $\Phi \,\&\, \Psi$. So that branch looks fine. (Indeed, that branch is exactly the same as the linear development of our actual conjunction rule, so this reasoning is the same as that on p. 99.)

But what of the right branch? Assume that some interpretation satisfies Ψ; does that guarantee that it satisfies $\Phi \,\&\, \Psi$? Certainly not. So if the right branch is our open branch, a completed tree using this rule may remain open, even if its root is unsatisfiable. Completeness will be violated. Let's construct a counterexample of that form. We want a tree that includes a conjunction, whose root is unsatisfiable, but whose right branch remains open. Notice that the right branch 'ignores' the first conjunct; this is a clue that a good way to construct a counterexample will be to locate the unsatisfiability within that first conjunct. Suppose, for example, that we let Φ itself stand for a contradiction. If so, any conjunction with Φ as a conjunct will be unsatisfiable. But if Ψ is not a contradiction, then a tree with root $\Phi \,\&\, \Psi$ will remain open. Let's develop a tree with this root: $(P \equiv \neg P) \,\&\, Q$. Note that this sentence has a contradictory first conjunct, and a contingent, atomic second conjunct.

$$(P \equiv \neg P)\,\&\,Q \checkmark$$

$$(P \equiv \neg P) \checkmark \qquad Q$$
$$Q \qquad\qquad \uparrow$$

$$P \qquad \neg P$$
$$\neg P \qquad \neg\neg P$$
$$\times \qquad \times$$
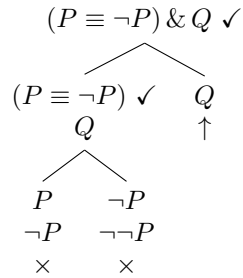
The left branch closes after we perform the unchanged biconditional rule on the contradictory first conjunct, but the right branch remains open. Since this is a completed tree with an open branch and an unsatisfiable root, it is a counterexample to completeness.

Remember that we are considering a modification to the tree system that uses a different conjunction rule. In this example I used $P \equiv \neg P$ as my Φ, which let me use the unchanged biconditional rule. But if I had used a contradiction that used a conjunction, like $P \,\&\, \neg P$, I would have had to have used the revised rule within the left branch too.

It is also possible to construct counterexamples using simpler sentences if you add to the root. Instead of introducing a contradictory conjunct, we could have simply added to the root, in a way that makes the root unsatisfiable, but leaves the right branch open. Suppose for instance we put both $P \,\&\, Q$ and $\neg P$ in the root. Then we'd have an unsatisfiable root, but the right branch would remain open:

$$P \,\&\, Q \checkmark$$
$$\neg P$$

$$P \qquad Q$$
$$Q \qquad \uparrow$$
$$\times$$
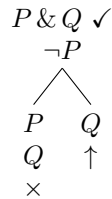
This too is a counterexample to completeness, given the rule in question. So we see here two different kinds of strategies for generating counterexamples to completeness.

Practice Exercises

★ **Part A** Following are possible modifications to our SL tree system. For each, imagine a system that is like the system laid out in this chapter, except for the indicated change. Would the modified tree system be sound? If so, explain how the proof given in this chapter would extend to a system with this rule; if not, give a tree that is a counterexample to the soundness of the modified system.

1. Change the rule for conjunctions to this rule:

$$\Phi \,\&\, \Psi$$
$$\Phi \qquad \Psi$$
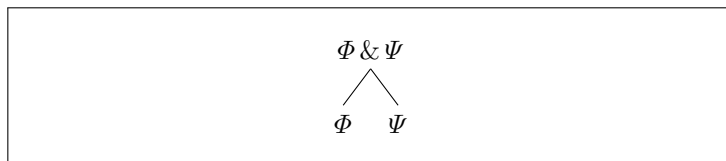
2. Change the rule for conjunctions to this rule:

$$\Phi \,\&\, \Psi$$
$$\Phi$$
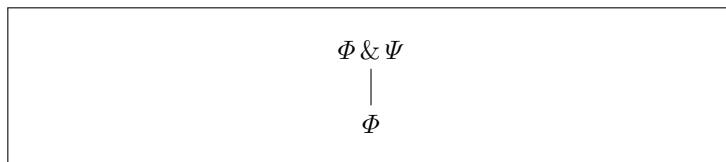
3. Change the rule for conjunctions to this rule:

$$\Phi \,\&\, \Psi$$
$$\Phi$$
$$\neg\Psi$$
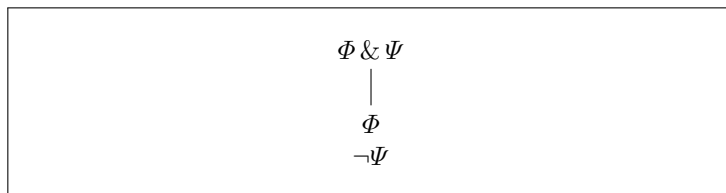
4. Change the rule for disjunctions to this rule:

$$\Phi \lor \Psi$$
$$\Phi$$
$$\Psi$$
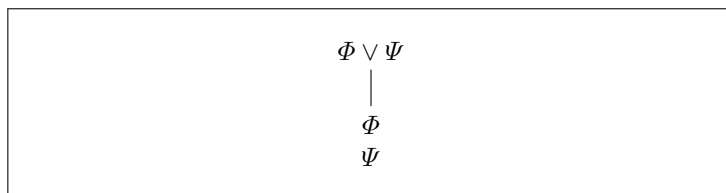
5. Change the rule for disjunctions to this rule:

$$\Phi \lor \Psi$$
$$\Phi \qquad \Psi \qquad \Phi \,\&\, \Psi$$
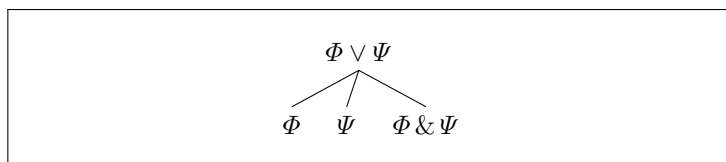
6. Change the rule for conditionals to this rule:

$$\Phi \supset \Psi$$

$$\neg\Phi \qquad \Psi \vee \Phi$$
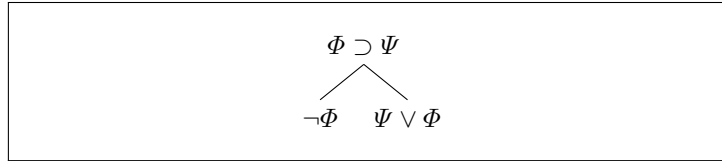
7. Change the rule for conditionals to this rule:

$$\Phi \supset \Psi$$

$$\neg\Phi \vee \Psi$$
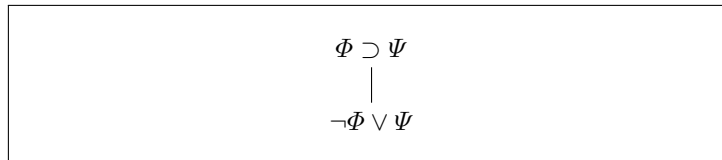
8. Change the rule for biconditionals to this rule:
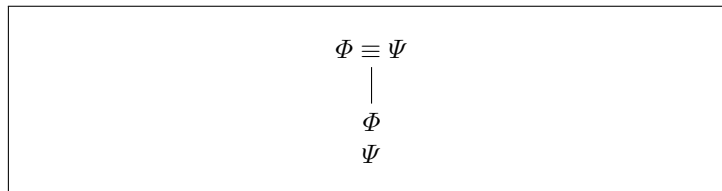
$$\Phi \equiv \Psi$$

$$\Phi$$
$$\Psi$$

9. Change the rule for disjunctions to this rule:
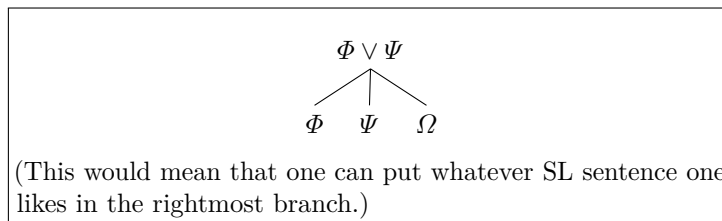
$$\Phi \vee \Psi$$

$$\Phi \qquad \Psi \qquad \Omega$$

(This would mean that one can put whatever SL sentence one likes in the rightmost branch.)

Part B For each of the rule modifications given in Part A, would the modified tree system be complete? If so, explain how the proof given in this chapter would extend to a system with this rule; if not, give a tree that is a counterexample to the completeness of the modified system.

Chapter 7

Natural Deduction Proofs in SL

This chapter introduces a different proof system in SL, separate from the tree method. The tree method has advantages and disadvantages. One advantage of trees is that, for the most part, they can be produced in a purely mechanical way — one need rely only on a rigorous application of a well-defined procedure; no 'flash of insight' is necessary. Another advantage is that, when a tree remains open, the tree method gives us a recipe for constructing an interpretation that satisfies the root. One disadvantage is that they do not always emphasize in an intuitive way *why* a conclusion follows from a set of premises; they show that something *must* be the case, on pain of contradiction, but they don't always demonstrate, in a way closely connected to natural reasoning, why some things follow from other things.

The NATURAL DEDUCTION system of this chapter will differ from the tree method in all of these respects. It is intended to model human reasoning in a closer way, illustrating the connections between various claims; consequently, working through a natural deduction proof requires a bit more insight and inspiration than a tree proof does.

Natural deduction proofs can be used to prove that an argument is valid; if an argument is invalid, our natural deduction system will not necessarily make that obvious. We won't have the equivalent to a completed open tree — a proof of invalidity. It will turn out that there is a natural deduction derivation corresponding to every valid argument in SL — i.e., like the tree method, this method is complete. (It is also sound.) But we won't offer a way to prove that there is no proof, and so natural deduction can't be used by itself to prove invalidity.

7.1 Natural deduction: the basic idea

The general idea of a natural deduction proof is simple. You begin by writing down the premises you're beginning with, each on its own numbered line. Then you add a new numbered line, adding sentences that you can demonstrate to follow logically from what you have already written. (A formal list of rules gives the legal options for how to develop the proof.) If, following the rules, you manage to derive the conclusion from the premises, then you've shown that the argument form from those premises to that conclusion is valid.

Formally, a PROOF is a numbered sequence of sentences. The first sentences of the sequence are assumptions; these are the premises of the argument. Every sentence later in the sequence is derived from earlier sentences by one of the rules of proof. The final sentence of the sequence is the conclusion of the argument.

Consider these two SL argument forms:

Modus Ponens: *Disjunctive Syllogism:*

$$P \supset Q$$
$$P$$
$$\therefore\ Q$$

$$P \vee Q$$
$$\neg P$$
$$\therefore\ Q$$

Both are valid; you could confirm this with a four-line truth table. Either would demonstrate that there is no interpretation satisfying both premises, while falsifying the conclusion. The truth table method does not distinguish between these argument forms; it simply shows that they are both valid. There is, however, an interesting and important difference between these two argument forms, gestured at by their labels. The first argument, *Modus Ponens*, has a distinctive syntactic form: its premises are a conditional and the antecedent of that conditional, and the conclusion is the consequent; the second has a different form.

The natural deduction method is based in the recognition of particular kinds of valid forms. They also correspond reasonably well to familiar forms of informal reasoning. If you know a conditional, and you also know its antecedent, it is easy to infer its consequent. (Imagine being sure that if I ate the chilli, I'll get sick, and also being sure that I ate the chilli. You will surely infer that I will get sick.) *Modus ponens* is the name of this kind of conditional reasoning, and there is a special rule for it in our natural deduction system.

7.2 Our first rule: Conditional Elimination (*modus ponens*)

Many different arguments demonstrate the *modus ponens* pattern; all of them are valid:

$P \supset \neg Q$	$\neg P \supset (A \equiv B)$	$(P \vee Q) \supset A$
P	$\neg P$	$P \vee Q$
$\therefore \neg Q$	$\therefore (A \equiv B)$	$\therefore A$

All of these arguments are similar in an important way to the *modus ponens* argument above, and different from the *disjunctive syllogism* one. But the truth table proof of the validity for each of these arguments will go the same way: it will simply produce a truth table, and show that there is no row where the premises are all true and the conclusion is false. The truth table method does not illuminate what the *modus ponens* inferences all have in common.

Another way of making this point is to observe that the method of truth tables does not clearly show *why* an argument is valid. If you were to do a 1024-line truth table for an argument that contains ten sentence letters, then you could check to see if there were any lines on which the premises were all true and the conclusion were false. If you did not see such a line, and you were sure that you made no mistakes in constructing the table, then you would know that the argument was valid. Yet you might not be able to say anything further about why this particular argument was a valid argument form.

The natural deduction system of this chapter will include an inference rule corresponding to *modus ponens*. Natural deduction proofs of these validities, then, will all share something important in common: they will use that inference rule. Here, in its formal presentation, is the rule:

$$
\begin{array}{r|l}
m & \Phi \supset \Psi \\[2ex]
n & \Phi \\[1ex]
\hline
 & \Psi \qquad\qquad \supset\!\mathrm{E}\ m, n
\end{array}
$$

As has been the case throughout this book, the Greek letters Φ and Ψ are variables that can represent any SL sentence. The m and n here are variables ranging over line numbers. What this rule says is that if you have a conditional $\Phi \supset \Psi$ on line m, and you also have Φ, the antecedent of that conditional, on line n, you can write the consequent Ψ on a new line.

The notation off to the right in the new line, "$\supset\!\mathrm{E}\ m, n$," is the justification for the new line. "$\supset\!\mathrm{E}$" stands for *conditional elimination*, which is our official name

for the *modus ponens* rule. (We call it an 'elimination' rule because it begins with a conditional statement, and ends up deriving a statement that is not a conditional.)

With this rule, we can prove that any of the *modus ponens* arguments mentioned above is valid. Here are proofs of two of them:

1	$P \supset \neg Q$			1	$(P \lor Q) \supset A$	
2	P			2	$P \lor Q$	
3	$\neg Q$	$\supset$E 1, 2		3	A	$\supset$E 1, 2

Notice that these two proofs share just the same structure. We start by listing the premises as lines 1 and 2. We include a horizontal line under those premises to indicate that subsequent lines will need to be derived via rules; so we apply the conditional elimination rule to get the conclusion. We justify that conclusion by citing conditional elimination, and lines 1 and 2.

One can produce more complicated proofs via the same rule. Let's prove that this SL argument form is valid:

$$A \supset B$$
$$B \supset C$$
$$C \supset [\neg P \equiv (Q \lor R)]$$
$$A$$
$$\therefore \ \neg P \equiv (Q \lor R)$$

We begin by writing our four premises on numbered lines:

1	$A \supset B$	
2	$B \supset C$	
3	$C \supset [\neg P \equiv (Q \lor R)]$	
4	A	want $\neg P \equiv (Q \lor R)$)

We also include the 'want' notation off to the right, indicating what conclusion we are attempting to establish. We complete the proof by deriving $\neg P \equiv (Q \lor R)$. We cannot use conditional elimination to get to that sentence directly from our premises. We do have, on line 3, a conditional with that sentence as a consequent, but we don't have C, the antecedent, on its own line. However, we can get there via two steps of our inference rule. From lines 1 and 4, we can get B, and then, from there and 2, we can get to C. Finally, a third instance of conditional elimination lets us derive the intended conclusion:

1	A	
2	$A \supset B$	
3	$B \supset C$	
4	$C \supset [\neg P \equiv (Q \vee R)]$	want $\neg P \equiv (Q \vee R))$
5	B	$\supset$E 1, 2
6	C	$\supset$E 3, 5
7	$\neg P \equiv (Q \vee R))$	$\supset$E 4, 6

Here are three things to notice about this proof.

First, every line after the premises needs to include a justification, citing a rule and the applicable previous line numbers. (So far conditional elimination is our only rule, but we will learn more very soon.) You can only write a new line if you have a rule that shows how it follows from previous lines.

Second, once you have derived something from the premises, that new line is available to help justify future lines. Once we derive line 5, for example, we're able to use it with line 2 to derive line 6.

Third, the conditional elimination rule requires that you have a conditional and its antecedent; it doesn't not matter what order they are listed in. (In other words, when the rule says you need to have a conditional on line m, and its antecedent on line n, it does not matter whether $m < n$.) It also does not matter whether they are on consecutive lines. So it is fine, for instance, that in justifying line 5, we cite a conditional on line 2, and its antecedent on line 1.

7.3 Exact matches

Conditional elimination, as well as all of our other natural deduction rules, are syntactically defined. That is to say, the application of the rules depends on the exact shape of the SL sentences in question. Here, again, is the formal statement of the rule:

m	$\Phi \supset \Psi$	
n	Φ	
	Ψ	$\supset$E m, n

It says that any time one has, on one line, a sentence made up of some sentence Φ, followed by the '$\supset$' symbol, followed by some sentence Ψ, where one also has

Φ on another line, one may derive Ψ. This is the only pattern of inference that this rule permits. Φ and Ψ can be any sentences of SL, but a line justified by conditional elimination must fit this pattern exactly. It is not enough that one can 'just see' that a given sentence follows via a similar pattern of inference.

For example, this is *not* a legal derivation in our system:

$$
\begin{array}{r|l}
1 & P \supset (A \,\&\, B) \\
2 & P \\ \hline
3 & B \qquad\qquad\qquad \supset\text{E } 1,\,2
\end{array}
$$

The conditional elimination rule requires that the new sentence derived be the consequent of the conditional cited. But in this example, B is not the consequent of $P \supset (A \,\&\, B)$ — $A \,\&\, B$ is. It is true that B obviously follows from $A \,\&\, B$, but the conditional elimination rule doesn't allow you to derive things just because they obviously follow. (Neither does any other rule in our formal system.) To derive B from these premises we'll need to use another rule. (In particular, we will want to use the conjunction elimination rule, given below.)

To check to make sure you are applying the rules correctly, one good heuristic is to think about whether you are relying on the rule itself, or on your intuitive understanding of the meanings of the symbols we use in SL. Your intuitive understanding is a good way to think about which rules to use, but to check to make sure you're using the rules properly, think about whether the rules could explain why it is permissible to extend the derivation in the exact way you're working with. Pretend, for instance, that you have no idea what the '$\supset$' symbol meant, but you know only that if you have two sentences joined by it on one line, and the first of those two sentences on another line, then you are allowed to copy down the second sentence exactly on a new line. This — and no more — is what the conditional elimination rule permits you to do. (This is what we mean when we say the rule is syntactically defined.)

7.4 Our second rule: *modus tollens*

Here is another rule one can use with conditionals:

$$
\begin{array}{r|l}
m & \Phi \supset \Psi \\
n & \neg\Psi \\ \hline
 & \neg\Phi \qquad\quad \text{MT } m,\,n
\end{array}
$$

If you have a conditional on one numbered line, and the negation of its consequent

on another line, you may derive the negation of its antecedent on a new line. We abbreviate the justification for this rule as 'MT' for *modus tollens.* If you know that if she found the treasure, she is happy, and you also know that she isn't happy, then you can very sensibly infer that she didn't find the treasure.

You may notice an asymmetry between the terms we use for *modus ponens* and *modus tollens*: in the former case, we use Conditional Elimination ($\supset$E) as its official name, but in the latter case, we call the rule Modus Tollens (MT). We'll explain in more detail why we use the labels in this way when we discuss 'basic' and 'derived' rules in §7.8 below. For now, the important thing is to understand how to use the rule.

Here is an example employing *Modus Tollens* several times over. We will prove that $\{A \supset B, B \supset C, C \supset D, \neg D\} \models \neg A$:

$$
\begin{array}{r|ll}
1 & A \supset B & \\
2 & B \supset C & \\
3 & C \supset D & \\
4 & \neg D & \text{want } \neg A \\
\hline
5 & \neg C & \text{MT 3, 4} \\
6 & \neg B & \text{MT 2, 5} \\
7 & \neg A & \text{MT 1, 6}
\end{array}
$$

At each of lines 5–7, we cite a conditional and the negation of its consequent to infer the negation of its antecedent.

7.5 Disjunction Elimination

Recall this argument form from the Introduction:

$$
\begin{array}{l}
P \lor Q \\
\neg P \\
\hline
\therefore Q
\end{array}
$$

We noted above that this is a different argument form from one that uses *modus ponens*; instead, it uses a kind of derivation that is specific to disjunction. From a disjunction like $P \lor Q$ alone, you can't conclude either disjunct, but you can conclude that *at least one* of the two disjuncts is true. So if you have also

established that one of the disjuncts is false — i.e., its negation is true — then you can conclude the other disjunct, as in the argument above.

This inference form is sometimes called 'Disjunctive Syllogism'. In our system, it will be our official Disjunction Elimination ($\vee$E) rule. If you have a disjunction and also the negation of one of its disjuncts, you may conclude the other disjunct.

m	$\Phi \vee \Psi$		m	$\Phi \vee \Psi$	
n	$\neg\Psi$		n	$\neg\Phi$	
	Φ	$\vee$E m, n		Ψ	$\vee$E m, n

We represent two different inference patterns here, because the rule allows you to conclude *either* disjunct from the negation of the other. If we'd only listed the left version of the rule above, then $\vee$E would've only permitted one to conclude the *first* disjunct from the negation of the *second* one, along with the disjunction. Our rule lets us work with either disjunction. (If you want to be very fussy about it, you could think of these as two different rules with a strong conceptual similarity that happen to have the same name.)

Now that we have several rules, we are in a position to see how they can interact to construct more interesting proofs. Consider for example this argument form:

$$\neg L \supset (J \vee L)$$
$$\neg L$$
$$\therefore\ J$$

The two premises match the requirements for Conditional Elimination (*modus ponens*). And once that is done, we will be in a position to use Disjunction Elimination to derive the desired conclusion:

1	$\neg L \supset (J \vee L)$	
2	$\neg L$	want J
3	$J \vee L$	$\supset$E 1, 2
4	J	$\vee$E 2, 3

By modus ponens, (1) and (2) entail $J \vee L$. Now from $J \vee L$ and $\neg L$, we can derive J, by Disjunction Elimination.

In this example, unlike the others we've seen so far, we used the premise on line 2 twice — notice that the 2 appears in the justification for both line 3 and line 4. There is no limit to how many times you can make use of a given line in a

natural deduction proof; once something is established, you can make use of it as often as you like.

7.6 Conjunction Introduction

Here is another rule. It is our Conjunction Introduction rule, which we abbreviate '&I':

$$
\begin{array}{c|l}
m & \Phi \\
n & \Psi \\
\hline
 & \Phi \,\&\, \Psi \qquad \&\,\text{I } m,\, n
\end{array}
$$

This rule says that if you have some sentence Φ, and you also have some sentence Ψ, you may derive their conjunction, $(\Phi \,\&\, \Psi)$. It is called Conjunction *Introduction* because it derives a conjunction from sentences that are not conjunctions.

Recall again that it is not a requirement either conjunct be an atom, or that m and n be consecutive lines, or that they appear in the order listed here. We require only that each line has been established somewhere above in the proof. If you have K on line 15 and L on line 8, you can prove $(K \,\&\, L)$ at some later point in the proof with the justification '&I 8, 15.'

7.7 Conjunction Elimination

We have just seen that Conjunction Introduction allows you derive a conjunction from a non-conjunction; Conjunction Elimination lets you do the converse. From a conjunction, you may derive either of its conjuncts. From $(A \,\&\, (P \lor Q))$, for instance, you may derive A, or you may derive $(P \lor Q)$. (Or you could even apply Conjunction Elimination twice and derive both.)

Because it lets you work on either conjunct, we formalize it twice over, much as we did with Disjunction Elimination. This will be our Conjunction Elimination rule, which we abbreviate '&E':

$$
\begin{array}{c|l}
m & \Phi \,\&\, \Psi \\
 & \Phi \qquad \&\,\text{E } m \\
 & \Psi \qquad \&\,\text{E } m
\end{array}
$$

The &E rule requires only one sentence, so we write one line number as the

justification for applying it.

Here is an example illustrating our two conjunction rules working together. Consider this argument.

$$[(A \vee B) \supset (C \vee D)] \,\&\, [(E \vee F) \supset (G \vee H)]$$
$$\therefore \; [(E \vee F) \supset (G \vee H)] \,\&\, [(A \vee B) \supset (C \vee D)]$$

The main logical operator in both the premise and conclusion is conjunction. Since conjunction is symmetric, the argument is obviously valid. The two conjunctions have the same two conjuncts, in the opposite order. In order to provide a proof, we begin by writing down the premise. After the premises, we draw a horizontal line — everything below this line must be justified by a rule of proof. So the beginning of the proof looks like this:

1 │ $[(A \vee B) \supset (C \vee D)] \,\&\, [(E \vee F) \supset (G \vee H)]$

From the premise, we can get each of the conjuncts by & E. The proof now looks like this:

1 │ $[(A \vee B) \supset (C \vee D)] \,\&\, [(E \vee F) \supset (G \vee H)]$
2 │ $[(A \vee B) \supset (C \vee D)]$ & E 1
3 │ $[(E \vee F) \supset (G \vee H)]$ & E 1

The rule & I requires that we have each of the conjuncts available somewhere in the proof. They can be separated from one another, and they can appear in any order. So by applying the & I rule to lines 3 and 2, we arrive at the desired conclusion. The finished proof looks like this:

1 │ $[(A \vee B) \supset (C \vee D)] \,\&\, [(E \vee F) \supset (G \vee H)]$
2 │ $[(A \vee B) \supset (C \vee D)]$ & E 1
3 │ $[(E \vee F) \supset (G \vee H)]$ & E 1
4 │ $[(E \vee F) \supset (G \vee H)] \,\&\, [(A \vee B) \supset (C \vee D)]$ & I 2, 3

This proof may not look terribly interesting or surprising, but it shows how we can use rules of proof together to demonstrate the validity of an argument form. Note also that using a truth table to show that this argument is valid would have required a staggering 256 lines, since there are eight sentence letters in the argument. A proof via trees would be less unwieldy than that, but it would be less simple and elegant than this one. (Constructing such a proof would be a good exercise for tree review.)

7.8 Basic and derived rules

We have so far introduced five rules: Conditional Elimination, Modus Tollens, Disjunction Elimination, Conjunction Elimination, and Conjunction Introduction. There are still more rules still to learn, but it is helpful first to pause and draw a distinction between different kinds of rules.

Many of our rules, we have seen, carry the name 'Elimination' or 'Introduction', along with the name of one of our SL connectives. In fact, every rule except *Modus Ponens* has had such a name. Such rules are the *basic* rules in our natural deduction system. The basic rules comprise an Introduction and an Elimination rule for each connective, plus one more rule. *Modus Tollens* is not a basic rule; we will call it a *derived* rule.

A derived rule is a non-basic rule whose validity we can derive using basic rules only. A very interesting fact about our natural deduction system is that the basic rules themselves are enough to derive every SL validity. Derived rules are helpful for making some proofs shorter or more intuitive, but one could prove anything provable without them.

We have already seen the introduction and elimination rules for conjunction, and the elimination rules for disjunction and conditionals. In the next several sections, we'll finish canvassing the basic rules, then say a bit more about *modus tollens* and other derived rules.

7.9 The Remaining Basic Rules

All of the rules introduced in this chapter are summarized in the Quick Reference guide at the end of this book.

7.9.1 Disjunction Introduction

If M is true, then $M \lor N$ must also be true. In general, the disjunction introduction rule ($\lor$I) allows us to derive a disjunction if we already have one of its two disjuncts:

$$
\begin{array}{l|ll}
m & \Phi & \\
\hline
 & \Phi \lor \Psi & \lor\text{I } m \\
 & \Psi \lor \Phi & \lor\text{I } m \\
\end{array}
$$

One can introduce a disjunct in either position — it can be the first disjunct or the second disjunct. Accordingly, both options are listed here. (One is not required to do both; you can just take whichever version you find most helpful.)

As always, Ψ can be *any* sentence whatsoever. So the following is a legitimate proof:

$$
\begin{array}{l|l}
1 & M \\ \hline
2 & M \vee ([(A \equiv B) \supset (C \,\&\, D)] \equiv [E \,\&\, F]) \qquad \vee\text{I } 1
\end{array}
$$

It may seem odd that just by knowing M we can derive a conclusion that includes sentences like A, B, and the rest — sentences that have nothing to do with M. Yet the conclusion follows immediately by $\vee$I. This is as it should be: The truth conditions for the disjunction mean that, if Φ is true, then $\Phi \vee \Psi$ is true regardless of what Ψ is. So the conclusion could not be false if the premise were true; the argument is valid.

7.9.2 Conditional Introduction

Consider this argument:

$$
\begin{array}{l}
R \vee F \\
\therefore\ \neg R \supset F
\end{array}
$$

The argument seems like it should be valid. (You can confirm this by examining the truth tables.) What should the conditional introduction rule be, such that we can draw this conclusion?

We begin the proof by writing down the premise of the argument, making a note of our intended conclusion, and drawing a horizontal line, like this:

$$
\begin{array}{l|l}
1 & R \vee F \qquad \text{want } \neg R \supset F
\end{array}
$$

If we had $\neg R$ as a further premise, we could derive F by the $\vee$E rule. But we do not have $\neg R$ as a premise of this argument, nor can we derive it directly from the premise we do have — so we cannot simply prove F. What we will do instead is start a *subproof*, a proof within the main proof. When we start a subproof, we draw another vertical line to indicate that we are no longer in the main proof. Then we write in an assumption for the subproof. This can be anything we want. Here, it will be helpful to assume $\neg R$. We want to show that, if we did assume that, we would be able to derive F. So we make a new assumption that $\neg R$, and give ourselves a note that we wish to derive F. Our proof now looks like this:

$$
\begin{array}{l|l}
1 & R \lor F \qquad \text{want } \lnot R \supset F \\
2 & \quad \boxed{\lnot R} \qquad \text{want } F
\end{array}
$$

It is important to emphasize that we are not claiming to have proven $\lnot R$ from the premise on line 1. We do not need to write in any justification for the assumption line of a subproof. (The 'want' is a note to ourself, not a justification.) The new vertical line indicates that an *assumption* is being made. You can think of the subproof as posing the question: What could we show *if* $\lnot R$ were true? We are trying to show that we could derive F. And indeed, we can:

$$
\begin{array}{l|l}
1 & R \lor F \qquad \text{want } \lnot R \supset F \\
2 & \quad \lnot R \qquad \text{want } F \\
3 & \quad F \qquad \lor\text{E } 1,2
\end{array}
$$

This has shown that *if* we had $\lnot R$ as a premise, *then* we could prove F. In effect, we have proven that F follows from $\lnot R$. This is indeed very close to demonstrating the conditional, $\lnot R \supset F$. This last step is just what the Conditional Introduction ($\supset$I) rule will allow us to perform. We close the subproof and derive $\lnot R \supset F$ in the main proof. Our final proof looks like this:

$$
\begin{array}{l|l}
1 & R \lor F \\
2 & \quad \lnot R \qquad \text{want } F \\
3 & \quad F \qquad \lor\text{E } 1,2 \\
4 & \lnot R \supset F \qquad \supset\text{I } 2\text{–}3
\end{array}
$$

The $\supset$I lets us DISCHARGE the assumption we'd been making, ending that vertical line. During lines (2) and (3), we were *assuming* that $\lnot R$; by the time we get to line (4), we are no longer making that assumption.

Notice that the justification for applying the $\supset$I rule is the entire subproof. That's why we justify it by reference to a range of lines, instead of a comma-separated list. Usually that will be more than just two lines.

It may seem as if the ability to assume anything at all in a subproof would lead to chaos: Does it allow you to prove any conclusion from any premises? The answer is no, it does not. Consider this proof schema:

$$
\begin{array}{l|l}
1 & \Phi \\
2 & \quad \Psi \\
3 & \quad \Psi \,\&\, \Phi \qquad \&\,\mathrm{I}\ 1,\,2 \\
\end{array}
$$

Does this show that one can conjoin any arbitrary sentence Ψ with premise Φ? After all, we've written $\Psi \,\&\, \Phi$ on a line of a proof that began with Φ, without violating any of the rules of our system. The reason this doesn't have that implication is the vertical line that still extends into line 3. That line indicates that the assumption made at line 2 is still in effect. When the vertical line for the subproof ends, the subproof is *closed.* In order to complete a proof, you must close all of the subproofs. The conclusion to be proved must not be 'blocked off' by a vertical line; it should be aligned with the premises.

In this example, there is no way to close the subproof and show that the conjunction follows from line 1 alone. One can only close a subproof via particular rules that allow you to do so; $\supset\mathrm{I}$ is one such rule; $\&\,\mathrm{I}$ does not close subproofs. One can't just close a subproof willy-nilly. Closing a subproof is called *discharging* the assumptions of that subproof. So we can put the point this way: You cannot complete a proof until you have discharged all of the assumptions other than the original premises of the argument.

Of course, it is legitimate to do this:

$$
\begin{array}{l|l}
1 & \Phi \\
2 & \quad \Psi \\
3 & \quad \Psi \,\&\, \Phi \qquad \&\,\mathrm{I}\ 1,\,2 \\
4 & \Psi \supset (\Psi \,\&\, \Phi) \qquad \supset\mathrm{I}\ 2\text{--}3 \\
\end{array}
$$

This should not seem so strange, though. The conclusion on line 4 really does follow from line 1. (Draw a truth table if you need convincing of this.)

Put in its general form, the $\supset\mathrm{I}$ rule looks like this:

$$
\begin{array}{l|l}
m & \quad \Phi \qquad (\text{want } \Psi) \\
n & \quad \Psi \\
 & \Phi \supset \Psi \qquad \supset\mathrm{I}\ m\text{--}n \\
\end{array}
$$

When we introduce a subproof, we typically write what we want to derive off to the right. This is just so that we do not forget why we started the subproof if it goes on for five or ten lines. There is no 'want' rule. It is a note to ourselves,

and not formally part of the proof.

Although it is always permissible to open a subproof with any assumption you please, there is some strategy involved in picking a useful assumption. Starting a subproof with a random assumption is a terrible strategy. It will just waste lines of the proof. In order to derive a conditional by the ⊃I rule, for instance, you must assume the antecedent of the conditional in a subproof.

The ⊃I rule also requires that the consequent of the conditional be the last line of the subproof. It is always permissible to close a subproof and discharge its assumptions, but it will not be helpful to do so until you get what you want. This is an illustration of the observation made above, that unlike the tree method, the natural deduction method requires some strategy and thinking ahead.

Never make an assumption without a plan for how to discharge it.

Here is another example of an argument form whose validity we can prove using the conditional rules.

$$P \supset Q$$
$$Q \supset R$$
$$\therefore P \supset R$$

We begin the proof by writing the two premises as assumptions. Since the main logical operator in the conclusion is a conditional, we can expect to use the ⊃I rule. For that, we need a subproof — so we write in the antecedent of the conditional as assumption of a subproof. We make a note that we are aiming for the consequent of that conditional:

1	$P \supset Q$	
2	$Q \supset R$	
3	$\quad P$	want R

We made P available by assuming it in a subproof, allowing us to use ⊃E on the first premise. This gives us Q, which allows us to use ⊃E on the second premise. Having derived R, we close the subproof. By assuming P we were able to prove R, so we apply the ⊃I rule and finish the proof.

$$
\begin{array}{ll}
1 & P \supset Q \\
2 & Q \supset R \\
3 & \quad\quad P \quad\quad\quad \text{want } R \\
4 & \quad\quad Q \quad\quad\quad \supset\text{E } 1,\, 3 \\
5 & \quad\quad R \quad\quad\quad \supset\text{E } 2,\, 4 \\
6 & P \supset R \quad\quad \supset\text{I } 3\text{--}5
\end{array}
$$

7.9.3 Biconditional Introduction

Biconditionals indicate that the two sides have the same truth value. One establishes a biconditional by establishing each direction of it as conditionals. To derive $W \equiv X$, for instance, you must establish both $W \supset X$ *and* $X \supset W$. Those conditionals may occur in either order; they need not be on consecutive lines. (Compare the shape of the &I rule.) Schematically, the rule works like this:

$$
\begin{array}{ll}
m & \Phi \supset \Psi \\
n & \Psi \supset \Phi \\
  & \Phi \equiv \Psi \quad\quad \equiv\text{I } m,\, n
\end{array}
$$

7.9.4 Biconditional Elimination

The biconditional elimination rule ($\equiv$E) is a generalized version of *modus ponens* ($\supset$E). If you have the left-hand subsentence of the biconditional, you can derive the right-hand subsentence. If you have the right-hand subsentence, you can derive the left-hand subsentence. This is the rule:

$$
\begin{array}{ll}\quad\quad\quad\quad
m & \Phi \equiv \Psi \\
n & \Phi \\
  & \Psi \quad\quad \equiv\text{E } m,\, n
\end{array}
\qquad\qquad
\begin{array}{ll}
m & \Phi \equiv \Psi \\
n & \Psi \\
  & \Phi \quad\quad \equiv\text{E } m,\, n
\end{array}
$$

As in the case of Disjunction Elimination, we include both versions under the same name, so that you don't need to worry about whether the side you already have is the left-hand side of the biconditional or the right-hand side. Whichever side it is, you may derive the other via biconditional elimination.

7.9.5 Negation Introduction and Elimination

Here is a simple mathematical argument in English:

> Assume there is some greatest natural number. Call it A.
> That number plus one is also a natural number.
> Obviously, $A + 1 > A$.
> So there is a natural number greater than A.
> This is impossible, since A is assumed to be the greatest natural number.
> ∴ There is no greatest natural number.

This kind of argument form is traditionally called a *reductio*. Its full Latin name is *reductio ad absurdum*, which means 'reduction to absurdity.' (You will also sometimes see proofs of this form described as 'indirect' proofs.) In a *reductio*, we assume something for the sake of argument — for example, that there is a greatest natural number. Then we show that the assumption leads to two contradictory sentences — for example, that A is the greatest natural number and that it is not. In this way, we show that the original assumption must have been false.

The basic rules for negation will allow for arguments like this. Like the Conditional Introduction rule, our negation rules allow us to make new assumptions with no justification — we draw a new vertical line, and a note to ourselves as to what we are trying to do. If we assume something and show that it leads to contradictory sentences, then we have proven the negation of the assumption. This is the negation introduction (¬I) rule:

$$
\begin{array}{lll}
m & \quad\big| \quad \Phi & \text{(for reductio)} \\[4pt]
n & \quad\big| \quad \Psi & \\[4pt]
o & \quad\big| \quad \neg\Psi & \\[4pt]
  & \neg\Phi & \text{¬I } m\text{–}n,\ m\text{–}o
\end{array}
$$

The ¬I rule discharges the assumption for *reductio*, concluding its negation, when it's shown that some sentence and its negation each follow from the assumption. It cites two (overlapping) ranges: a subproof from the assumption to some sentence Ψ, and a subproof from that same assumption to $\neg\Psi$. We write 'for reductio' to the right of the assumption, as a note to ourselves, a reminder of why we started the subproof. It is not formally part of the proof, but it is helpful for thinking clearly it and planning ahead.

To see how the rule works, suppose we want to prove an instance of the law of non-contradiction: $\neg(G \,\&\, \neg G)$. We can prove this without any premises by immediately starting a subproof. We want to apply ¬I to the subproof, so we

assume $(G \& \neg G)$. We then get an explicit contradiction by $\&$ E. The proof looks like this:

$$
\begin{array}{ll}
1 & \quad \begin{array}{|l} G \& \neg G \end{array} \quad \text{for reductio} \\
2 & \quad \begin{array}{|l} G \end{array} \quad \& \text{E 1} \\
3 & \quad \begin{array}{|l} \neg G \end{array} \quad \& \text{E 1} \\
4 & \neg(G \& \neg G) \quad \neg\text{I 1–2, 1–3}
\end{array}
$$

The $\neg$E rule will work in much the same way. If we assume $\neg\varPhi$ and show that it leads to a sentence and its negation, we have effectively proven $\varPhi$. So the rule looks like this:

$$
\begin{array}{ll}
m & \quad \begin{array}{|l} \neg\varPhi \end{array} \quad \text{for reductio} \\
n & \quad \begin{array}{|l} \varPsi \end{array} \\
o & \quad \begin{array}{|l} \neg\varPsi \end{array} \\
p & \varPhi \quad \neg\text{E } m\text{–}n,\ m\text{–}o
\end{array}
$$

7.9.6 Reiteration

In addition to the Introduction and Elimination rules for each logical operator, we will also have one more basic rule: Reiteration (R). If you already have shown something in the course of a proof, the reiteration rule allows you to repeat it on a new line. Put formally:

$$
\begin{array}{ll}
m & \quad \begin{array}{|l} \varPhi \end{array} \\
& \quad \begin{array}{|l} \varPhi \end{array} \quad \text{R } m
\end{array}
$$

Obviously, the reiteration rule will not allow us to show anything *new*. For that, we will need to use rules. Still, Reiteration is a handy rule to have available. Here, for instance, is a natural deduction proof using Reiteration to help with a Negation Introduction:

1	P	
2	$Q \supset \neg P$	want $\neg Q$
3	$\quad Q$	for reductio
4	$\quad \neg P$	$\supset$E 2, 3
5	$\quad P$	R 1
6	$\neg Q$	$\neg$I 3–4, 3–5

Negation Introduction requires that one show that some sentence and its negation are both derivable from the assumption for *reductio*. In this case, we establish that P and $\neg P$ both follow from the assumption that Q. In the case of $\neg P$, we can get it directly via conditional elimination; the easiest way to get the P that we need is simply to reiterate it from line 1.

You may have noticed that there is another way to prove $\neg Q$ from these premises — one may do it in a single step via *modus tollens*. Recall that *modus tollens* is not one of our *basic* rules. It is a *derived* rule that lets us take certain shortcuts. But anything that can be proven with *modus tollens* can be proven with the basic rules listed above. In fact, the proof we have just seen, using Negation Introduction and Reiteration, provides a template. Anything you could get to with one step via *modus tollens*, you could also get to in four steps via Negation Introduction, Conditional Elimination, and Reiteration.

7.10 Derived rules

A DERIVED RULE is a rule of proof that does not make any new proofs possible. Anything that can be proven with a derived rule can be proven without it. You can think of a short proof using a derived rule as shorthand for a longer proof that uses only the basic rules.

7.10.1 Modus tollens

We have been discussing *modus tollens* throughout this chapter, and in §7.9.6 we showed how, in the case of one simple proof using *modus tollens*, we can instead give a slightly longer proof using only basic rules. In fact, we can prove this much more generally, via this proof schema:

$$
\begin{array}{ll}
1 & \Phi \\
2 & \Psi \supset \neg\Phi \qquad \text{want } \neg\Psi \\
3 & \quad \Psi \qquad \text{for reductio} \\
4 & \quad \neg\Phi \qquad \supset\text{E } 2, 3 \\
5 & \quad \Phi \qquad \text{R } 1 \\
6 & \neg\Psi \qquad \neg\text{I } 3\text{--}4, 3\text{--}5
\end{array}
$$

As always, Φ and Ψ meta-variables. They are not symbols of SL, but stand-ins for arbitrary sentences of SL. So this is not, strictly speaking, a proof in SL. It is more like a recipe; no matter what sentences you want to use *modus tollens* on, if that rule would allow it, this pattern of proof will get you to the same place using basic rules only. This means that the rule is not really necessary; we could prove everything we want to prove without it.

Nevertheless, it is a convenient shortcut. We add it to our list of derived rules.

7.10.2 Dilemma

Here is the Dilemma rule:

$$
\begin{array}{ll}
m & \Phi \vee \Psi \\
n & \Phi \supset \Omega \\
o & \Psi \supset \Omega \\
& \Omega \qquad \text{DIL } m, n, o
\end{array}
$$

It might not be immediately obvious that this is a valid inference rule, but if you think about it a minute, you may be able to see why it is a good rule. If you know that two conditionals are true, and they have the same consequent, and you also know that one of the two antecedents is true, then whichever of those two disjunct is true, will license a *modus ponens* inference to the conclusion.

For example, suppose you know all of the following:

> If it is raining, the car is wet.
> If it is snowing, the car is wet.
> It is raining or it is snowing.

From these premises, you can definitely establish that the car is wet. This is the form that the Dilemma rule captures. We use the label 'dilemma' to convey

the idea that, whichever way you pick, you'll be stuck with the same conclusion. ("You're damned if you do, and you're damned if you don't. You do or don't. Therefore you are damned.")

The Dilemma rule is also derivable from the basic rules. The proof is a bit more complicated, but here it is in schematic form:

1	$\Phi \vee \Psi$	
2	$\Phi \supset \Omega$	
3	$\Psi \supset \Omega$	want Ω
4	$\neg\Omega$	for reductio
5	Φ	for reductio
6	Ω	$\supset$E 2, 5
7	$\neg\Omega$	R 4
8	$\neg\Phi$	$\neg$I 5–6, 5–7
9	Ψ	for reductio
10	Ω	$\supset$E 3, 9
11	$\neg\Omega$	R 4
12	Ψ	$\vee$E 1, 8
13	$\neg\Psi$	$\neg$I 9–10, 9–11
14	Ω	$\neg$E 4–12, 4–13

To understand this proof, think about the broad outline of it. Ultimately, we derive Ω via Negation Elimination; that's why we assumed $\neg\Omega$ on line 4, and worked our way to a contradiction on lines 12–13. The internal lines are subproofs to the intermediate conclusions Ψ and $\neg\Psi$.

So as in the case of *modus tollens*, Dilemma doesn't allow us to prove anything we couldn't prove via basic rules. Anytime you wanted to use the Dilemma rule, you could always take ten extra lines and prove the same thing in basic rules. But it's a useful shorthand to include on our list of derived rules.

7.10.3 Hypothetical Syllogism

We also add hypothetical syllogism (HS) as a derived rule:

$$
\begin{array}{c|ll}
m & \Phi \supset \Psi & \\
n & \Psi \supset \Omega & \\
& \Phi \supset \Omega & \text{HS } m,\, n
\end{array}
$$

We have already given a proof of one instance of Hypothetical Syllogism it on p. 122. The general proof is just the same, but with that P, Q, and R replaced with Φ, Ψ, and Ω, respectively.

7.11 Rules of replacement

Consider how you would prove this argument form valid: $F \supset (G \,\&\, H) \therefore F \supset G$

Perhaps it is tempting to write down the premise and apply the $\&\,\mathrm{E}$ rule to the conjunction $(G \,\&\, H)$. This is impermissible, however, because the basic rules of proof can only be applied to whole sentences. In order to use $\&\,\mathrm{E}$, we need to get the conjunction $(G \,\&\, H)$ on a line by itself. Here is a proof:

$$
\begin{array}{c|ll}
1 & F \supset (G \,\&\, H) & \\
2 & \quad F & \text{want } G \\
3 & \quad G \,\&\, H & \supset\!\mathrm{E}\ 1,\, 2 \\
4 & \quad G & \&\,\mathrm{E}\ 3 \\
5 & F \supset G & \supset\!\mathrm{I}\ 2\text{--}4
\end{array}
$$

The rules we have seen so far must apply to wffs that are on a proof line by themselves. We will now introduce some derived rules that may be applied to wffs that are parts of more complex sentences. These are called RULES OF REPLACEMENT, because they can be used to replace part of a sentence with a logically equivalent expression. One simple rule of replacement is Commutativity (abbreviated Comm), which says that we can swap the order of conjuncts in a conjunction or the order of disjuncts in a disjunction. We define the rule this way:

$$
\begin{aligned}
(\Phi \,\&\, \Psi) &\Longleftrightarrow (\Psi \,\&\, \Phi) \\
(\Phi \vee \Psi) &\Longleftrightarrow (\Psi \vee \Phi) \\
(\Phi \equiv \Psi) &\Longleftrightarrow (\Psi \equiv \Phi) \quad \text{Comm}
\end{aligned}
$$

The double arrow means that you can take a subformula on one side of the arrow and replace it with the subformula on the other side. The arrow is double-headed

because rules of replacement work in both directions. And replacement rules — unlike all the rules we've seen so far — can be applied to wffs that are part of more complex sentences. They don't need to be on their own line.

Consider this argument: $(M \vee P) \supset (P \,\&\, M)$, $\therefore$ $(P \vee M) \supset (M \,\&\, P)$

It is possible to give a proof of this using only the basic rules, but it will be long and inconvenient. With the Comm rule, we can provide a proof easily:

1	$(M \vee P) \supset (P \,\&\, M)$	
2	$(P \vee M) \supset (P \,\&\, M)$	Comm 1
3	$(P \vee M) \supset (M \,\&\, P)$	Comm 2

(We need to apply the rule twice, because each application allows one transformation. We transformed the antecedent first, then the consequent.)

Another rule of replacement is double negation (DN). With the DN rule, you can remove or insert a pair of negations for any wff in a line, even if it isn't the whole line. This is the rule:

$$\neg\neg\varPhi \Longleftrightarrow \varPhi \quad \text{DN}$$

Two more replacement rules are called De Morgan's Laws, named for the 19th-century British logician August De Morgan. (Although De Morgan did formalize and publish these laws, many others discussed them before him.) The rules capture useful relations between negation, conjunction, and disjunction. Here are the rules, which we abbreviate DeM:

$$\neg(\varPhi \vee \varPsi) \Longleftrightarrow (\neg\varPhi \,\&\, \neg\varPsi)$$
$$\neg(\varPhi \,\&\, \varPsi) \Longleftrightarrow (\neg\varPhi \vee \neg\varPsi) \quad \text{DeM}$$

As we have seen, $\varPhi \supset \varPsi$ is equivalent to $\neg\varPhi \vee \varPsi$. A further replacement rule captures this equivalence. We abbreviate the rule MC, for 'material conditional.' It takes two forms:

$$(\varPhi \supset \varPsi) \Longleftrightarrow (\neg\varPhi \vee \varPsi)$$
$$(\varPhi \vee \varPsi) \Longleftrightarrow (\neg\varPhi \supset \varPsi) \quad \text{MC}$$

Now consider this argument: $\neg(P \supset Q)$, $\therefore$ $P \,\&\, \neg Q$

As always, we could prove this argument valid using only the basic rules. With rules of replacement, though, the proof is much simpler:

1	$\neg(P \supset Q)$	
2	$\neg(\neg P \vee Q)$	MC 1
3	$\neg\neg P \,\&\, \neg Q$	DeM 2
4	$P \,\&\, \neg Q$	DN 3

A final replacement rule captures the relation between conditionals and biconditionals. We will call this rule biconditional exchange and abbreviate it $\equiv$ex.

$$[(\Phi \supset \Psi) \,\&\, (\Psi \supset \Phi)] \Longleftrightarrow (\Phi \equiv \Psi) \quad \equiv\text{ex}$$

7.12 Proof strategy

There is no simple recipe for proofs, and there is no substitute for practice. Here, though, are some rules of thumb and strategies to keep in mind.

Work backwards from what you want.   The ultimate goal is to derive the conclusion. Look at the conclusion and ask what the introduction rule is for its main logical operator. This gives you an idea of what should happen *just before* the last line of the proof. Then you can treat this line as if it were your goal. Ask what you could do to derive this new goal.

For example: If your conclusion is a conditional $\Phi \supset \Psi$, plan to use the $\supset$I rule. This requires starting a subproof in which you assume Φ. In the subproof, you want to derive Ψ.

Work forwards from what you have.   When you are starting a proof, look at the premises; later, look at the sentences that you have derived so far. Think about the elimination rules for the main operators of these sentences. These will tell you what your options are.

For example: If you have a conditional $\Phi \supset \Psi$, and you also have Φ, $\supset$E is a pretty natural choice.

For a short proof, you might be able to eliminate the premises and introduce the conclusion. A long proof is formally just a number of short proofs linked together, so you can fill the gap by alternately working back from the conclusion and forward from the premises.

Change what you are looking at.   Replacement rules can often make your life easier. If a proof seems impossible, try out some different substitutions.

For example: It is often difficult to prove a disjunction using the basic rules. If you want to show $\Phi \vee \Psi$, it is often easier to show $\neg\Phi \supset \Psi$ and use the MC rule.

Some replacement rules should become second nature. If you see a negated disjunction, for instance, you should immediately think of DeMorgan's rule.

Do not forget indirect proof.   If you cannot find a way to show something directly, try assuming its negation.

Remember that most proofs can be done either indirectly or directly. One way might be easier — or perhaps one sparks your imagination more than the other — but either one is formally legitimate.

Repeat as necessary.   Once you have decided how you might be able to get to the conclusion, ask what you might be able to do with the premises. Then consider the target sentences again and ask how you might reach them.

Persist.   Try different things. If one approach fails, then try something else.

7.13 Proof-theoretic concepts

As we did in our discussion of trees, we will again use the symbol '⊢' to indicate provability. Provability is relative to a proof system, so the meaning of the '⊢' symbol featured in this chapter should be distinguished from the one we used for trees. When necessary, we can specify the single turnstile with reference to the proof system in question, letting '$\vdash_T$' stand for provability in the tree system, and '$\vdash_{ND}$' stand for provability in this natural deduction system. For the most part in this chapter, though, we'll be interested in natural deduction, so unless it is specified otherwise, you can understand '⊢' to mean '$\vdash_{ND}$'.

The double turnstile symbol '⊨', remains unchanged. It stands for semantic entailment, as described in ch. 4.

When we write $\{\Phi_1, \Phi_2, \ldots\} \vdash_{ND} \Psi$, this means that it is possible to give a natural deduction derivation of Ψ from premises $\Phi_1, \Phi_2, \ldots$. With just one premise, we leave out the curly brackets, so $\Phi \vdash \Psi$ means that there is a proof of Ψ with Φ as a premise. Naturally, $\vdash \Phi$ means that there is a proof of Φ that has no premises. You can think of it as shorthand for $\varnothing \vdash \Phi$.

For notational completeness, we can understand $\mathcal{X} \vdash \bot$ to mean that from $\mathcal{X}$, we could prove an arbitrary contradiction. In other words, $\mathcal{X}$ is an inconsistent set.

Logical proofs are sometimes called *derivations*. So $\Phi \vdash \Psi$ can be read as 'Ψ is derivable from Φ.'

A THEOREM is a sentence that is derivable without any premises; i.e., Φ is a theorem if and only if $\vdash \Phi$.

It is not too hard to show that something is a theorem — you just have to give a proof of it. How could you show that something is *not* a theorem? If its negation is a theorem, then you could provide a proof. For example, it is easy to prove $\neg(P \mathbin{\&} \neg P)$, which shows that $(P \mathbin{\&} \neg P)$ cannot be a theorem. For a sentence that is neither a theorem nor the negation of a theorem, however, there is no easy way to show this. You would have to demonstrate not just that certain proof strategies fail, but that no proof is possible. Even if you fail in trying to prove a sentence in a thousand different ways, perhaps the proof is just too long and complex for you to make out. As we've emphasized already, this is a difference between our natural deduction system and the tree method. (For just the same reason, the natural deduction system doesn't provide a straightforward way to demonstrate that an argument form is invalid.)

Two sentences Φ and Ψ are PROVABLY EQUIVALENT if and only if each can be derived from the other; i.e., $\Phi \vdash \Psi$ and $\Psi \vdash \Phi$.

It is relatively easy to show that two sentences are provably equivalent — it just requires a pair of proofs. Showing that sentences are *not* provably equivalent would be much harder. It would be just as hard as showing that a sentence is not a theorem. (In fact, these problems are interchangeable. Can you think of a sentence that would be a theorem if and only if Φ and Ψ were provably equivalent?)

The set of sentences $\{\Phi_1, \Phi_2, \ldots\}$ is PROVABLY INCONSISTENT if and only if contradictory sentences are derivable from it; i.e., for some sentence Ψ, $\{\Phi_1, \Phi_2, \ldots\} \vdash \Psi$ and $\{\Phi_1, \Phi_2, \ldots\} \vdash \neg\Psi$. This is equivalent to $\{\Phi_1, \Phi_2, \ldots\} \vdash \bot$.

It is easy to show that a set is provably inconsistent: You just need to assume the sentences in the set and prove a contradiction. Showing that a set is *not* provably inconsistent will be much harder. It would require more than just providing a proof or two; it would require showing that proofs of a certain kind are *impossible*.

7.14 Proofs and models

As you might already suspect, there is a connection between *theorems* and *tautologies*.

There is a formal way of showing that a sentence is a theorem: Prove it. For each line, we can check to see if that line follows by the cited rule. It may be hard to produce a twenty line proof, but it is not so hard to check each line of the proof and confirm that it is legitimate — and if each line of the proof individually is legitimate, then the whole proof is legitimate. Showing that a sentence is a tautology, though, requires reasoning in English about all possible models. There is no formal way of checking to see if the reasoning is sound. Given a choice between showing that a sentence is a theorem and showing that it is a tautology, it would be easier to show that it is a theorem.

By contrast, there is no formal way of showing that a sentence is *not* a theorem. We would need to reason in English about all possible proofs. Yet there is a formal method for showing that a sentence is not a tautology. We need only construct a model in which the sentence is false. Given a choice between showing that a sentence is not a theorem and showing that it is not a tautology, it would be easier to show that it is not a tautology.

Fortunately, a sentence is a theorem if and only if it is a tautology. If we provide a proof of $\vdash \Phi$ and thus show that it is a theorem, it follows that Φ is a tautology; i.e., $\models \Phi$. Similarly, if we construct a model in which Φ is false and thus show that it is not a tautology, it follows that Φ is not a theorem.

In general, $\Phi \vdash \Psi$ if and only if $\Phi \models \Psi$. As such:

> ▷ An argument is *valid* if and only if *the conclusion is derivable from the premises.*

> ▷ Two sentences are *logically equivalent* if and only if they are *provably equivalent.*

> ▷ A set of sentences is *consistent* if and only if it is *not provably inconsistent.*

You can pick and choose when to think in terms of proofs and when to think in terms of models, doing whichever is easier for a given task. Table 7.1 summarizes when it is best to give proofs and when it is best to give models.

In this way, proofs and models give us a versatile toolkit for working with arguments. If we can translate an argument into SL, then we can measure its logical weight in a purely formal way. If it is deductively valid, we can give a formal proof; if it is invalid, we can provide a formal counterexample.

	YES	NO
Is Φ a tautology?	prove $\vdash \Phi$	give a model in which Φ is false
Is Φ a contradiction?	prove $\vdash \neg\Phi$	give a model in which Φ is true
Is Φ contingent?	give a model in which Φ is true and another in which Φ is false	prove $\vdash \Phi$ or $\vdash \neg\Phi$
Are Φ and Ψ equivalent?	prove $\Phi \vdash \Psi$ and $\Psi \vdash \Phi$	give a model in which Φ and Ψ have different truth values
Is the set $\mathbb{A}$ consistent?	give a model in which all the sentences in $\mathbb{A}$ are true	taking the sentences in $\mathbb{A}$, prove Ψ and $\neg\Psi$
Is the argument 'Φ, Ψ, ... $\therefore$ Ω' valid?	prove $\Phi, \Psi, \ldots \vdash \Omega$	give a model in which $\{\Phi, \Psi, \ldots\}$ is satisfied and Ω is falsified

Table 7.1: Sometimes it is easier to show something by providing proofs than it is by providing models. Sometimes it is the other way round. It depends on what you are trying to show.

7.15 Soundness and completeness

Chapter 6 considered the soundness and completeness of the tree method at length; it proved that this method was both sound ($\mathcal{X} \vdash_T \Phi$ only if $\mathcal{X} \models \Phi$) and complete ($\mathcal{X} \models \Phi$ only if $\mathcal{X} \vdash_T \Phi$). We said in the last section that $\Phi \vdash_{ND} \Psi$ if and only if $\Phi \models \Psi$; this is equivalent to saying that our natural deduction system, too, is sound and complete. Given the soundness and completeness of the tree method, this also means that our two proof systems are equivalent in the sense that anything provable in one is also provable in the other ($\mathcal{X} \vdash_T \Phi$) iff ($\mathcal{X} \vdash_{ND} \Phi$).

How can we know that our natural deduction method is sound? A proof system is SOUND if there are no derivations corresponding to invalid arguments. Demonstrating that the proof system is sound would require showing that any possible proof in our system is the proof of a valid argument. There is a fairly simple way of approaching this in a step-wise fashion. If using the &E rule on the last line of a proof could never change a valid argument into an invalid one, then using the rule many times could not make an argument invalid. Similarly, if using the &E and ∨E rules individually on the last line of a proof could never change a valid argument into an invalid one, then using them in combination could not either.

The strategy is to show for every rule of inference that it alone could not make a valid argument into an invalid one. It follows that the rules used in combination would not make a valid argument invalid. Since a proof is just a series of lines, each justified by a rule of inference, this would show that every provable argument is valid.

Consider, for example, the &I rule. Suppose we use it to add $\Phi \,\&\, \Psi$ to a valid argument. In order for the rule to apply, Φ and Ψ must already be available in the proof. Since the argument so far is valid, Φ and Ψ are either premises of the argument or valid consequences of the premises. As such, any model in which the premises are true must be a model in which Φ and Ψ are true. According to the definition of TRUTH IN SL, this means that $\Phi \,\&\, \Psi$ is also true in such a model. Therefore, $\Phi \,\&\, \Psi$ validly follows from the premises. This means that using the &I rule to extend a valid proof produces another valid proof.

In order to show that the proof system is sound, we would need to show this for the other inference rules. Since the derived rules are consequences of the basic rules, it would suffice to provide similar arguments for the 16 other basic rules. The reasoning is extremely similar to that given in the soundness proof for trees in the previous chapter. We will not go through it in detail here.

What of completeness? Why think that *every* valid argument is an argument that can be proven in our natural deduction system? That is, why think that $\Phi \models \Psi$ implies $\Phi \vdash \Psi$? Our system *is* also complete, but the completeness proof for natural deduction is a bit more complex than the completeness proof for trees. (In the case of trees, we had a mechanical method that was guaranteed to find proofs if they exist; we have seen no such method here, which makes proving these general results harder.) This proof is beyond the scope of this book.

The important point is that, happily, the proof system for SL is both sound and complete. Consequently, we may freely use this natural deduction method to draw conclusions about models in SL.

Summary of definitions

▷ A sentence Φ is a THEOREM if and only if $\vdash \Phi$.

▷ Two sentences Φ and Ψ are PROVABLY EQUIVALENT if and only if $\Phi \vdash \Psi$ and $\Psi \vdash \Phi$.

▷ $\{\Phi_1, \Phi_2, \ldots\}$ is PROVABLY INCONSISTENT if and only if, for some sentence Ψ, $\{\Phi_1, \Phi_2, \ldots\} \vdash (\Psi \,\&\, \neg\Psi)$.

Practice Exercises

⋆ **Part A** Provide a justification (rule and line numbers) for each line of proof that requires one.

1	$W \supset \neg B$
2	$A \& W$
3	$B \vee (J \& K)$
4	W
5	$\neg B$
6	$J \& K$
7	K

1	$L \equiv \neg O$
2	$L \vee \neg O$
3	$\quad \neg L$
4	$\quad \neg O$
5	$\quad L$
6	$\quad \neg L$
7	L

1	$Z \supset (C \& \neg N)$
2	$\neg Z \supset (N \& \neg C)$
3	$\quad \neg (N \vee C)$
4	$\quad \neg N \& \neg C$
5	$\quad\quad Z$
6	$\quad\quad C \& \neg N$
7	$\quad\quad C$
8	$\quad\quad \neg C$
9	$\quad \neg Z$
10	$\quad N \& \neg C$
11	$\quad N$
12	$\quad \neg N$
13	$N \vee C$

⋆ **Part B** Give a proof for each argument in SL.

1. $K \& L, \therefore K \equiv L$
2. $A \supset (B \supset C), \therefore (A \& B) \supset C$
3. $P \& (Q \vee R), P \supset \neg R, \therefore Q \vee E$
4. $(C \& D) \vee E, \therefore E \vee D$
5. $\neg F \supset G, F \supset H, \therefore G \vee H$
6. $(X \& Y) \vee (X \& Z), \neg(X \& D), D \vee M \therefore M$

Part C Give a proof for each argument in SL.

1. $Q \supset (Q \& \neg Q), \therefore \neg Q$
2. $J \supset \neg J, \therefore \neg J$

   3. $E \vee F$, $F \vee G$, $\neg F$, $\therefore$ $E \,\&\, G$
   4. $A \equiv B$, $B \equiv C$, $\therefore$ $A \equiv C$
   5. $M \vee (N \supset M)$, $\therefore$ $\neg M \supset \neg N$
   6. $S \equiv T$, $\therefore$ $S \equiv (T \vee S)$
   7. $(M \vee N) \,\&\, (O \vee P)$, $N \supset P$, $\neg P$, $\therefore$ $M \,\&\, O$
   8. $(Z \,\&\, K) \vee (K \,\&\, M)$, $K \supset D$, $\therefore$ D

⋆ **Part D** Show that each of the following sentences is a theorem in SL.

   1. $O \supset O$
   2. $N \vee \neg N$
   3. $\neg(P \,\&\, \neg P)$
   4. $\neg(A \supset \neg C) \supset (A \supset C)$
   5. $J \equiv [J \vee (L \,\&\, \neg L)]$

Part E Show that each of the following pairs of sentences are provably equivalent in SL.

   1. $\neg\neg\neg\neg G$, G
   2. $T \supset S$, $\neg S \supset \neg T$
   3. $R \equiv E$, $E \equiv R$
   4. $\neg G \equiv H$, $\neg(G \equiv H)$
   5. $U \supset I$, $\neg(U \,\&\, \neg I)$

⋆ **Part F** Provide proofs to show each of the following.

   1. $M \,\&\, (\neg N \supset \neg M) \vdash (N \,\&\, M) \vee \neg M$
   2. $\{C \supset (E \,\&\, G), \neg C \supset G\} \vdash G$
   3. $\{(Z \,\&\, K) \equiv (Y \,\&\, M), D \,\&\, (D \supset M)\} \vdash Y \supset Z$
   4. $\{(W \vee X) \vee (Y \vee Z), X \supset Y, \neg Z\} \vdash W \vee Y$

Part G For the following, provide proofs using only the basic rules. The proofs will be longer than proofs of the same claims would be using the derived rules.

   1. Show that MT is a legitimate derived rule. Using only the basic rules, prove the following: $\Phi \supset \Psi$, $\neg\Psi$, $\therefore$ $\neg\Phi$
   2. Show that Comm is a legitimate rule for the biconditional. Using only the basic rules, prove that $\Phi \equiv \Psi$ and $\Psi \equiv \Phi$ are equivalent.
   3. Using only the basic rules, prove the following instance of DeMorgan's Laws: $(\neg A \,\&\, \neg B)$, $\therefore$ $\neg(A \vee B)$
   4. Show that $\equiv$ex is a legitimate derived rule. Using only the basic rules, prove that $D \equiv E$ and $(D \supset E) \,\&\, (E \supset D)$ are equivalent.

Part H

1. If you know that $\Phi \vdash \Psi$, what can you say about $(\Phi \,\&\, \Omega) \vdash \Psi$? Explain your answer.
2. If you know that $\Phi \vdash \Psi$, what can you say about $(\Phi \lor \Omega) \vdash \Psi$? Explain your answer.

Chapter 8

Quantified logic

This chapter introduces a logical language called QL. It is a version of *quantified logic*, because it allows for quantifiers like *all* and *some*. Quantified logic is also sometimes called *predicate logic*, because the basic units of the language are predicates and terms.

8.1 From sentences to predicates

Consider the following argument, which is valid in English:

> If everyone knows logic, then either no one will be confused or everyone will. Everyone will be confused only if we try to believe a contradiction. Everyone knows logic.
> ∴ If we don't try to believe a contradiction, then no one will be confused.

In order to symbolize this in SL, we will need a symbolization key.

> **L:** Everyone knows logic.
> **N:** No one will be confused.
> **E:** Everyone will be confused.
> **B:** We try to believe a contradiction.

Notice that N and E are both about people being confused, but they are two separate sentence letters. We can't replace E with $\neg N$. Why not? $\neg N$ means 'It is not the case that no one will be confused.' This would be the case if even

one person were confused, so it is a long way from saying that *everyone* will be confused.

Because we have separate sentence letters for *N* and *E*, however, our formalization does not encode any connection between the two. They are just two atomic sentences which might be true or false independently. It is impossible for it to be the case that both no one and everyone was confused. As sentences of SL, however, there is a truth-value assignment for which *N* and *E* are both true. This is a limitation of the descriptive power of SL. Some features of English sentences are not preserved in SL. Our new language, QL, will preserve more of this structure.

Expressions like 'no one', 'everyone', and 'anyone' are called *quantifiers*. By translating *N* and *E* as separate atomic sentences, we leave out the *quantifier structure* of the sentences. In the example we've been discussing, the quantifier structure is not terribly important. The argument is valid without reference to it. As such, we can safely ignore it. To see this, we translate the argument to SL:

$$L \supset (N \vee E)$$
$$E \supset B$$
$$L$$
$$\therefore \neg B \supset N$$

This is a valid argument form in SL. (You can construct a truth table, a tree, or a natural deduction proof to confirm this.)

Now consider another argument. This one is also valid in English.

> Willard is a logician. All logicians wear funny hats.
> ∴ Willard wears a funny hat.

To symbolize it in SL, we define a symbolization key:

L: Willard is a logician.
A: All logicians wear funny hats.
F: Willard wears a funny hat.

Now we symbolize the argument:

$$L$$
$$A$$
$$\therefore F$$

This is pretty obviously an *invalid* SL form. Nevertheless, this is clearly a valid argument in English. It's impossible for the premises to be true without the conclusion also being true. But the SL symbolization leaves out all quantificational structure in virtue of which it is valid. The sentence 'All logicians wear funny hats' says something specific about both logicians and hat-wearing. By not translating this structure, treating the whole sentence as an atom, we lose the connection between Willard's being a logician and Willard's wearing a hat.

Some arguments with quantifier structure can be captured in SL, like the first example, even though SL ignores the quantifier structure. Other arguments' validity cannot be captured in SL, like the second example. Notice that the problem is not that we have made a mistake while symbolizing the second argument. We gave a perfectly appropriate translation; indeed, these are the best symbolizations we can give for these arguments *in SL*.

If an English argument is properly translated into a form that is valid in SL, that argument is valid, even if it involves quantifiers. But if it does not have a valid SL form, that doesn't mean the English argument is invalid; valid arguments can have invalid forms. It just means that SL doesn't *show* that the argument is valid. This will be the case when the argument's quantifier structure plays an important role in its validity.

Similarly, if a sentence with quantifiers comes out as a *tautology in SL*, then the English sentence is logically true. If it comes out as *contingent in SL*, then this might be because of the structure of the quantifiers that gets removed when we translate into the formal language.

In order to symbolize arguments that rely on quantifier structure, we need to develop a different, richer, logical language. We will call this language 'quantified logic', or QL.

8.2 Building blocks of QL

Our first key notion in QL will be PREDICATES. A predicate is analogous to an English description — an expression like 'is a dog' or 'has black fur'. Such descriptions are not sentences on their own. They are neither true nor false. They are descriptions that apply to some objects, but not to others. In order to be true or false, we need to specify an object: Who or what is it that is said to be a dog, or to have black fur? In SL, we have no way to give meaningful translations to English terms that are not full sentences. In QL, we will.

The details of this will be explained in the rest of the chapter, but here is the basic idea: In QL, we will represent predicates with capital letters. For instance, we might let *D* stand for '_____ is a dog.' We will use lower-case letters as

the names of specific things. For instance, we might let *b* stand for Bertie. The expression *Db* will be a sentence in QL. Given this symbolization, *Db* is a translation of the sentence 'Bertie is a dog.'

In order to represent quantifier structure, we will also have symbols that represent quantifiers. For instance, '∃' will mean 'There is some_____.' So to say that there is a dog, we can write ∃*xDx*; that is: There is some *x* such that *x* is a dog.

That will come later. We start by defining singular terms and predicates.

8.3 Singular terms

In English, a SINGULAR TERM is a word or phrase that refers to a *specific* person, place, or thing. The word 'dog' is not a singular term, because it is a general term that could apply to many individual animals. The phrase 'Jonathan's dog Mezzo' is a singular term, because it refers to a specific little poodle mix. She is black, and soft, and wonderful.

A PROPER NAME is a singular term that picks out an individual directly. The name 'Emerson' is a proper name, and the name alone does not tell you anything about Emerson. Of course, some names are traditionally given to boys, and others are traditionally given to girls. If 'Jack Hathaway' is used as a singular term, you might guess that it refers to a man. However, the name does not necessarily mean that the person referred to is a man — or even that the creature referred to is a person. Jack might be a giraffe for all you could tell just from the name. There is a great deal of philosophical action surrounding this issue, but the important point here is that a name is a singular term because it picks out a single, specific individual.

Other singular terms more obviously convey information about the thing to which they refer. For instance, you can tell without being told anything further that 'Jonathan's dog Mezzo' is a singular term that refers to a dog. A DEFINITE DESCRIPTION picks out an individual by means of a unique description. In English, definite descriptions are often phrases of the form 'the such-and-so.' They refer to *the* specific thing that matches the given description. For example, 'the tallest member of Monty Python' and 'the first emperor of China' are definite descriptions. A description that does not pick out a specific individual is not a definite description. 'A member of Monty Python' and 'an emperor of China' are not definite descriptions. We'll discuss definite descriptions in more detail in Chapter 12.

In English, the specification of a singular term may depend on context; 'Willard' means a specific person and not just someone named Willard; 'P.D. Magnus' as a logical singular term means the original author of this textbook, not the other

person who has the same name. We live with this kind of ambiguity in English, but it is important to keep in mind that singular terms in QL must refer to just one specific thing.

In QL, we will symbolize singular terms with lower-case letters a through w. We can add subscripts if we want to use some letter more than once. So $a, b, c, \ldots w, a_1, f_{32}, j_{390}$, and m_{12} are all terms in QL.

Singular terms are called CONSTANTS because they pick out specific individuals. Note that x, y, and z are not constants in QL. They will be VARIABLES, letters which do not stand for any specific thing. We will need them when we introduce quantifiers.

8.4 Predicates

Simple one-place predicates are properties of individuals. They are things you can say about an object. Here are some one-place predicates:

> '_____ is a dog'
> '_____ is a member of Monty Python'
> '_____ 's favourite ramen place is in Gastown'
> 'An anvil was dropped from a very high height onto _____ 's head'

Predicates like these are called ONE-PLACE or MONADIC, because there is only one blank to fill in. A one-place predicate and a singular term combine to make a sentence.

Other predicates are about the *relation* between two things. For instance:

> '_____ is bigger than _____'
> '_____ is to the left of _____'
> '_____ owes money to _____'

These are TWO-PLACE or DYADIC predicates, because they need to be filled in with two terms in order to make a sentence.

In general, you can think about predicates as schematic sentences that need to be filled out with some number of terms. Conversely, you can start with sentences and make predicates out of them by removing terms. Consider the sentence, 'Buchanan Tower is North of Barber but South of Buchanan E.' By removing a singular term, we can recognize this sentence as using any of three different monadic predicates:

_____is North of Barber but South of Buchanan E.

Buchanan Tower is North of _____but South of Buchanan E.

Buchanan Tower is North of Barber but South of _____.

By removing two singular terms, we can recognize three different dyadic predicates:

Buchanan Tower is North of _____but South of _____.

_____is North of Barber but South of _____.

_____is North of _____but South of Buchanan E.

By removing all three singular terms, we can recognize one THREE-PLACE or TRIADIC predicate:

_____is North of _____but South of _____.

If we are translating this sentence into QL, should we translate it with a one-, two-, or three-place predicate? It depends on what we want to be able to say. If we're only interested in discussing the location of buildings relative to Barber, then the generality of the three-place predicate is unnecessary. If we only want to discuss whether buildings are North of Barber and South of BUCH E, a one-place predicate will be enough.

In general, we can have predicates with as many places as we need. Predicates with more than one place are called POLYADIC. Predicates with n places, for some number n, are called N-PLACE or N-ADIC. You can make an n-adic predicate by replacing n names in any sentence with blanks. You can even have a 0-place predicate — this would be the result of replacing 0 names in a sentence with blanks. In other words, a 0-place predicate is just a sentence.

In QL, we symbolize predicates with capital letters A through Z, with or without subscripts. When we give a symbolization key for predicates, we will not use blanks; instead, we will use variables. By convention, constants are listed at the end of the key. So we might write a key that looks like this:

Ax: x is angry.
Hx: x is happy.
T$_1$xy: x is at least as tall as y.
T$_2$xy: x is at least as tough as y.
Bxyz: y is between x and z.
d: Donald
g: Gregor
m: Marybeth

We can symbolize sentences that use any combination of these predicates and terms. For example:

1. Donald is angry.
2. If Donald is angry, then so are Gregor and Marybeth.
3. Marybeth is at least as tall and as tough as Gregor.
4. Donald is shorter than Gregor.
5. Gregor is between Donald and Marybeth.

Sentence 1 is straightforward: *Ad*. The 'x' in the key entry '*Ax*' is just a placeholder; we replace it with other terms when translating.

Sentence 2 can be paraphrased as, 'If *Ad*, then *Ag* and *Am*.' QL has all the truth-functional connectives of SL, so we translate this as $Ad \supset (Ag \,\&\, Am)$.

Sentence 3 can be translated as $T_1 mg \,\&\, T_2 mg$.

Sentence 4 might seem as if it requires a new predicate. If we only needed to symbolize this sentence, we could define a predicate like *Sxy* to mean 'x is shorter than y.' However, this would ignore the logical connection between 'shorter' and 'taller.' Considered only as symbols of QL, there is no connection between S and T_1. They might mean anything at all. Instead of introducing a new predicate, we paraphrase sentence 4 using predicates already in our key: 'It is not the case that Donald is as tall or taller than Gregor.' We can translate it as $\neg T_1 dg$.

Sentence 5 requires that we pay careful attention to the order of terms in the key. It becomes *Bdgm*.

8.5   Quantifiers

We are now ready to introduce quantifiers. Quantifiers (unlike names and predicates) are a new kind of logical connective; like conjunction, negation, etc., they can govern the interpretation of a QL sentence. Consider these English sentences:

6. Everyone is happy.
7. Everyone is at least as tough as Donald.
8. Someone is angry.

It might be tempting to translate sentence 6 as $Hd \,\&\, Hg \,\&\, Hm$. Yet this would only say that Donald, Gregor, and Marybeth are happy. We want to say that *everyone* is happy, even if we have not defined a constant to name them. In order to do this, we introduce the '∀' symbol. This is called the UNIVERSAL QUANTIFIER.

A quantifier is placed in front of the formula that it binds. It always comes along with an associated variable. Typically that formula will include the same variable. We can translate sentence 6 as $\forall x Hx$. Paraphrased partially into English, this means 'For all x, x is happy.'

We call $\forall x$ an *x-quantifier*. The formula that follows the quantifier is called the *scope* of the quantifier. We will give a formal definition of scope later, but intuitively it is the part of the sentence that the quantifier quantifies over. In $\forall x Hx$, the scope of the universal quantifier is Hx.

Sentence 7 can be paraphrased as, 'For all x, x is at least as tough as Donald.' This translates as $\forall x T_2 xd$.

In these quantified sentences, the variable x is serving as a kind of placeholder. The expression $\forall x$ means that you can pick anyone and put them in as x. There is no special reason to use x rather than some other variable. The sentence $\forall x Hx$ means exactly the same thing as $\forall y Hy$, $\forall z Hz$, and $\forall x_5 Hx_5$.

To translate sentence 8, we introduce another new symbol: the EXISTENTIAL QUANTIFIER, ∃. Like the universal quantifier, the existential quantifier requires a variable. Sentence 8 can be translated as $\exists x Ax$. This means that there is some x which is angry. More precisely, it means that there is *at least one* angry person. Once again, the variable is a kind of placeholder; we could just as easily have translated sentence 8 as $\exists z Az$.

Consider these further sentences:

9. No one is angry.
10. There is someone who is not happy.
11. Not everyone is happy.

Sentence 9 can be paraphrased as, 'It is not the case that someone is angry.' This can be translated using negation and an existential quantifier: $\neg \exists x Ax$. Yet sentence 9 could also be paraphrased as, 'Everyone is not angry.' With this in mind, it can be translated using negation and a universal quantifier: $\forall x \neg Ax$.

Both of these are acceptable translations, because they are logically equivalent. The critical thing is whether the negation comes before or after the quantifier.

In general, $\forall x \Phi$ is logically equivalent to $\neg \exists x \neg \Phi$. This means that any sentence which can be symbolized with a universal quantifier can be symbolized with an existential quantifier, and vice versa. One translation might seem more natural than the other, but there is no logical difference in translating with one quantifier rather than the other. For some sentences, it will simply be a matter of taste.

Sentence 10 is most naturally paraphrased as, 'There is some x such that x is not happy.' This becomes $\exists x \neg H x$. Equivalently, we could write $\neg \forall x H x$.

Sentence 11 is most naturally translated as $\neg \forall x H x$. This is logically equivalent to sentence 10 and so could also be translated as $\exists x \neg H x$.

8.6 Universe of discourse

Given the symbolization key we have been using, $\forall x H x$ means 'Everyone is happy.' Who is included in this *everyone*? When we use sentences like this in English, we usually do not mean everyone now alive on the Earth. We certainly do not mean everyone who was ever alive or who will ever live. We mean something more modest: everyone in the building, everyone in the class, or everyone in the room.

In order to eliminate this ambiguity, we will need to specify a UNIVERSE OF DISCOURSE — abbreviated UD. The UD is the set of things that we are talking about. So if we want to talk about people in Chicago, we define the UD to be people in Chicago. We write this at the beginning of the symbolization key, like this:

UD: people in Chicago

The quantifiers *range over* the universe of discourse. Given this UD, $\forall x$ means 'Everyone in Chicago' and $\exists x$ means 'Someone in Chicago.' Each constant names some member of the UD, so we can only use this UD with the symbolization key above if Donald, Gregor, and Marybeth are all in Chicago. If we want to talk about people in places besides Chicago, then we need to include those people in the UD.

In QL, the UD must be *non-empty*; that is, it must include at least one thing. It is possible to construct formal languages that allow for empty UDs, but this introduces complications; such languages are beyond the scope of this book.

Even allowing for a UD with just one member can produce some strange results.

Suppose we have this as a symbolization key:

UD: the Eiffel Tower
Px: x is in Paris.

The sentence $\forall x P x$ might be paraphrased in English as 'Everything is in Paris.' Yet that would be misleading. It means that everything *in the UD* is in Paris. This UD contains only the Eiffel Tower, so with this symbolization key $\forall x P x$ just means that the Eiffel Tower is in Paris. We will rarely work with such bizarre domains as this.

It is a rule in QL that each constant will pick out exactly one member of the UD. (There is no rule prohibiting multiple different constants from referring to the same member of the UD.)

8.7 Translating to QL

We now have the basic pieces of QL. Translating more complicated sentences will only be a matter of knowing the right way to combine predicates, constants, quantifiers, variables, and sentential connectives. Consider these sentences:

12. Every coin in my pocket is a loonie.
13. Some coin on the table is a dime.
14. Not all the coins on the table are loonies.
15. None of the coins in my pocket are dimes.

In providing a symbolization key, we need to specify a UD. Since we are talking about coins in my pocket and on the table, the UD must at least contain all of those coins. Since we are not talking about anything besides coins, we let the UD be all coins. Since we are not talking about any specific coins, we do not need to define any constants. So we define this key:

UD: all coins
Px: x is in my pocket.
Tx: x is on the table.
Lx: x is a loonie.
Dx: x is a dime.

Sentence 12 is most naturally translated with a universal quantifier. The universal quantifier says something about everything in the UD, not just about the coins in my pocket. Sentence 12 means that, for any coin, *if* that coin is in my pocket, *then* it is a loonie. So we can translate it as $\forall x (P x \supset L x)$.

Since sentence 12 is about coins that are both in my pocket *and* that are loonies, it might be tempting to translate it using a conjunction. However, the sentence $\forall x(Px \& Lx)$ would mean that everything in the UD is both in my pocket and a loonie: All the coins that exist are loonies in my pocket. This would be nice, but it means something very different than sentence 12.

Sentence 13 is most naturally translated with an existential quantifier. It says that there is some coin which is both on the table and which is a dime. So we can translate it as $\exists x(Tx \& Dx)$.

Notice that we needed to use a conditional with the universal quantifier, but we used a conjunction with the existential quantifier. This is a common pattern. What would it mean to write $\exists x(Tx \supset Dx)$? Probably not what you think. It means that there is some member of the UD which would satisfy the subformula; roughly speaking, there is some name α such that $(T\alpha \supset D\alpha)$ is true. In SL, $\Phi \supset \Psi$ is logically equivalent to $\neg\Phi \vee \Psi$, and this will also hold in QL. So $\exists x(Tx \supset Dx)$ is true if there is some α such that $(\neg T\alpha \vee D\alpha)$; i.e., it is true if some coin is *either* not on the table *or* is a dime. Of course there is a coin that is not on the table — there are coins in lots of other places. So $\exists x(Tx \supset Dx)$ makes an extremely weak claim. A conditional will usually be the natural connective to use with a universal quantifier, but a conditional within the scope of an existential quantifier can do very strange things. It's a pretty good rule of thumb that you shouldn't be putting conditionals in the scope of existential quantifiers. This is pretty much never a good translation of any natural English sentence.

Sentence 14 can be paraphrased as, 'It is not the case that every coin on the table is a loonie.' So we can translate it as $\neg\forall x(Tx \supset Lx)$. You might look at sentence 14 and paraphrase it instead as, 'Some coin on the table is not a loonie.' You would then translate it as $\exists x(Tx \& \neg Lx)$. Although it is probably not obvious, these two translations are logically equivalent. (This is due to the logical equivalence between $\neg\forall x\Phi$ and $\exists x\neg\Phi$, along with the equivalence between $\neg(\Phi \supset \Psi)$ and $\Phi \& \neg\Psi$. We'll be able to prove this later.)

Sentence 15 can be paraphrased as, 'It is not the case that there is some dime in my pocket.' This can be translated as $\neg\exists x(Px \& Dx)$. It might also be paraphrased as, 'Everything in my pocket is a non-dime,' and then could be translated as $\forall x(Px \supset \neg Dx)$. Again the two translations are logically equivalent. Both are correct translations of sentence 15.

We can now translate the argument from p. 133, the one that motivated the need for quantifiers:

> Willard is a logician. All logicians wear funny hats.
> ∴ Willard wears a funny hat.

UD: people

Lx: x is a logician.
Fx: x wears a funny hat.
w: Willard

Translating, we get:

$$Lw$$
$$\forall x(Lx \supset Fx)$$
$$\therefore \ Fw$$

This captures the structure that was left out of the SL translation of this argument, and this is a valid argument in QL.

8.8 Empty predicates

A predicate need not apply to anything in the UD. A predicate that applies to nothing in the UD is called an EMPTY predicate.

Suppose we want to symbolize these two sentences:

16. Every monkey knows sign language.
17. Some monkey knows sign language.

It is possible to write the symbolization key for these sentences in this way:

UD: animals
Mx: x is a monkey.
Sx: x knows sign language.

Sentence 16 can now be translated as $\forall x(Mx \supset Sx)$.

Sentence 17 becomes $\exists x(Mx \,\&\, Sx)$.

It is tempting to say that sentence 16 entails sentence 17; that is: if every monkey knows sign language, then it must be that some monkey knows sign language. However, the entailment does not hold in QL. It is possible for the sentence $\forall x(Mx \supset Sx)$ to be true even though the sentence $\exists x(Mx \,\&\, Sx)$ is false.

How can this be? The answer comes from considering whether these sentences would be true or false *if there were no monkeys.*

> ▷ A UD must have *at least* one member.
>
> ▷ A predicate may apply to some, all, or no members of the UD.
>
> ▷ A constant must pick out *exactly* one member of the UD.
>
> A member of the UD may be picked out by one constant, many constants, or none at all.

We have defined ∀ and ∃ in such a way that ∀*Φ* is equivalent to ¬∃¬*Φ*. As such, the universal quantifier doesn't involve the existence of anything — only non-existence. If sentence 16 is true, then there are *no* monkeys who don't know sign language. If there were no monkeys, then $\forall x(Mx \supset Sx)$ would be true and $\exists x(Mx \,\&\, Sx)$ would be false.

A second reason to allow empty predicates is that we want to be able to say things like, 'I do not know if there are any monkeys, but any monkeys that there are know sign language.' That is, we want to be able to have predicates that do not (or might not) refer to anything.

Third, consider: $\forall x(Px \supset Px)$. This should be a tautology. But if sentence 16 implied sentence 17, then this would imply $\exists x(Px \,\&\, Px)$. It would become a logical truth that for any predicate there is something that satisfies that predicate.

What happens if we add an empty predicate R to the interpretation above? For example, we might define Rx to mean 'x is a refrigerator.' Now the sentence $\forall x(Rx \supset Mx)$ will be true. This is counterintuitive, since we do not want to say that there are a whole bunch of refrigerator monkeys. It is important to remember, though, that $\forall x(Rx \supset Mx)$ means that any member of the UD which is a refrigerator is a monkey. Since the UD is animals, there are no refrigerators in the UD and so the sentence is trivially true.

If you were actually translating the sentence 'All refrigerators are monkeys', then you would want to include appliances in the UD. Then the predicate R would not be empty and the sentence $\forall x(Rx \supset Mx)$ would be false.

8.9 Picking a universe of discourse

The appropriate symbolization of an English language sentence in QL will depend on the symbolization key. In some ways, this is obvious: It matters whether Dx means 'x is dainty' or 'x is dangerous.' The meaning of sentences in QL also

depends on the UD.

Let Rx mean 'x is a rose,' let Tx mean 'x has a thorn,' and consider this sentence:

    18. Every rose has a thorn.

It is tempting to say that sentence 18 should be translated as $\forall x(Rx \supset Tx)$. If the UD contains all roses, that would be correct. Yet if the UD is merely *things on my kitchen table*, then $\forall x(Rx \supset Tx)$ would only mean that every rose on my kitchen table has a thorn. If there are no roses on my kitchen table, the sentence would be trivially true.

The universal quantifier only ranges over members of the UD, so we need to include all roses in the UD in order to translate sentence 18. We have two options. First, we can restrict the UD to include all roses but *only* roses. Then sentence 18 becomes $\forall x Tx$. This means that everything in the UD has a thorn; since the UD just is the set of roses, this means that every rose has a thorn. This option can save us trouble if every sentence that we want to translate using the symbolization key is about roses.

Second, we can let the UD contain things besides roses: rhododendrons, rats, rifles, and whatall else. Then sentence 18 must be $\forall x(Rx \supset Tx)$.

If we wanted the universal quantifier to mean *every* thing, without restriction, then we might try to specify a UD that contains everything. But this notion is somewhat obscure. Does 'everything' include things that have only been imagined, like fictional characters? On the one hand, we want to be able to symbolize arguments about Hamlet or Sherlock Holmes. So we need to have the option of including fictional characters in the UD. On the other hand, we never need to talk about every thing that does not exist. That might not even make sense. There are philosophical issues here that we will not try to address. We can avoid these difficulties by always specifying the UD. For example, if we mean to talk about plants, people, and cities, then the UD might be 'living things and places.'

Suppose that we want to translate sentence 18 and, with the same symbolization key, translate these sentences:

    19. Esmerelda has a rose in her hair.
    20. Everyone is cross with Esmerelda.

We need a UD that includes roses (so that we can symbolize sentence 18) and a UD that includes people (so we can translate sentence 19–20.) Here is a suitable key:

UD: people and plants
Px: x is a person.
Rx: x is a rose.
Tx: x has a thorn.
Cxy: x is cross with y.
Hxy: x has y in their hair.
e: Esmerelda

Since we do not have a predicate that means '...has a rose in her hair', translating sentence 19 will require paraphrasing. The sentence says that there is a rose in Esmerelda's hair; that is, there is something which is both a rose and is in Esmerelda's hair. So we get: $\exists x(Rx \,\&\, Hex)$.

It is tempting to translate sentence 20 as $\forall x Cxe$. Unfortunately, this would mean that every member of the UD is cross with Esmerelda — both people and plants. It would mean, for instance, that the rose in Esmerelda's hair is cross with her. Of course, sentence 20 does not mean that.

'Everyone' means every person, not every member of the UD. So we can paraphrase sentence 20 as, 'Every person is cross with Esmerelda.' We know how to translate sentences like this: $\forall x(Px \supset Cxe)$.

In general, the universal quantifier can be used to mean 'everyone' if the UD contains only people. If there are people and other things in the UD, then 'everyone' must be treated as 'every person'.

8.10 Translating pronouns

When translating to QL, it is important to understand the structure of the sentences you want to translate. What matters is the final translation in QL, and sometimes you will be able to move from an English language sentence directly to a sentence of QL. Other times, it helps to paraphrase the sentence one or more times. Each successive paraphrase should move from the original sentence closer to something that you can translate directly into QL.

For the next several examples, we will use this symbolization key:

UD: people
Gx: x can play guitar.
Rx: x is a rock star.
c: Chris
l: Lemmy

Now consider these sentences:

21. If Chris can play guitar, then they are a rock star.
22. If a person can play guitar, then they are a rock star.

Sentence 21 and sentence 22 have the same words in the consequent ('... they are a rock star'), but they cannot be translated in the same way. It helps to paraphrase the original sentences, replacing pronouns with explicit references.

Sentence 21 can be paraphrased as, 'If Chris can play guitar, then *Chris* is a rockstar.' The word 'they' in sentence 21 is being used to refer to a specific individual, Chris. This can obviously be translated as $Gc \supset Rc$.

Sentence 22 must be paraphrased differently: 'If a person can play guitar, then *that person* is a rock star.' The pronoun 'they' here is not about any particular person, so we need a variable. Translating halfway, we can paraphrase the sentence as, 'For any person x, if x can play guitar, then x is a rock star.' Now this can be translated as $\forall x(Gx \supset Rx)$. This is the same as, 'Everyone who can play guitar is a rock star.'

Consider these further sentences:

23. If anyone can play guitar, then Lemmy can.
24. If anyone can play guitar, then they are a rock star.

These two sentences have the same antecedent ('If anyone can play guitar...'), but they have different logical structures.

Sentence 23 can be paraphrased, 'If someone can play guitar, then Lemmy can play guitar.' The antecedent and consequent are separate sentences, so it can be symbolized with a conditional as the main logical operator: $\exists xGx \supset Gl$.

Sentence 24 can be paraphrased, 'For anyone, if that one can play guitar, then that one is a rock star.' It would be a mistake to symbolize this with an existential quantifier, because it is talking about everybody. The sentence is equivalent to 'All guitar players are rock stars.' It is best translated as $\forall x(Gx \supset Rx)$.

The English words 'any' and 'anyone' should typically be translated using quantifiers. As these two examples show, they sometimes call for an existential quantifier (as in sentence 23) and sometimes for a universal quantifier (as in sentence 24). If you have a hard time determining which is required, paraphrase the sentence with an English language sentence that uses words besides 'any' or 'anyone.'

8.11 Quantifiers and scope

In the sentence $\exists x G x \supset G l$, the scope of the existential quantifier is the expression Gx. Would it matter if the scope of the quantifier were the whole sentence? That is, does the sentence $\exists x (Gx \supset Gl)$ mean something different?

With the key given above, $\exists x G x \supset G l$ means that if there is some guitarist, then Lemmy is a guitarist. $\exists x (Gx \supset Gl)$ would mean that there is some person such that if that person were a guitarist, then Lemmy would be a guitarist. Recall that the conditional here is a material conditional; the conditional is true any time the antecedent is false. Let the constant p denote the author of this book, someone who is certainly not a guitarist. The sentence $Gp \supset Gl$ is true because Gp is false. Since someone (namely p) satisfies the sentence, then $\exists x (Gx \supset Gl)$ is true. The sentence is true because there is a non-guitarist, regardless of Lemmy's skill with the guitar.

Something strange happened when we changed the scope of the quantifier, because the conditional in QL is a material conditional. In order to keep the meaning the same, we would have to change the quantifier: $\exists x G x \supset G l$ means the same thing as $\forall x (Gx \supset Gl)$, and $\exists x (Gx \supset Gl)$ means the same thing as $\forall x G x \supset G l$.

Note that quantifiers count as logical connectives, so one can sensibly ask whether the main connective of a given sentence is a quantifier or something else. (Calling it a 'connective' can be slightly confusing, since, unlike connectives like conjunction and disjunction, it doesn't literally *connect* two sentences. Quantifiers are like negations in this respect — each does count as a connective.) If the scope of the quantifier is the entire sentence, then that quantifier is the main connective, as in $\forall x (Gx \supset Gl)$. If the scope of the quantifier is limited to a subsentence, then that quantifier is not the main connective. For example, in $\exists x G x \supset G l$, the main quantifier is the conditional; the existential is part of the antecedent.

8.12 Ambiguous predicates

Suppose we just want to translate this sentence:

25. Adina is a skilled surgeon.

Let the UD be people, let Kx mean 'x is a skilled surgeon', and let a mean Adina. Sentence 25 is simply Ka.

Suppose instead that we want to translate this argument:

The hospital will only hire a skilled surgeon. All surgeons are greedy. Billy is a surgeon, but is not skilled. Therefore, Billy is greedy, but the hospital will not hire him.

We need to distinguish being a *skilled surgeon* from merely being a *surgeon.* So we define this symbolization key:

UD: people
Gx: x is greedy.
Hx: The hospital will hire x.
Rx: x is a surgeon.
Kx: x is skilled.
b: Billy

Now the argument can be translated in this way:

$$\forall x \big[\neg(Rx \,\&\, Kx) \supset \neg Hx\big]$$
$$\forall x(Rx \supset Gx)$$
$$Rb \,\&\, \neg Kb$$
$$\therefore\ Gb \,\&\, \neg Hb$$

Next suppose that we want to translate this argument:

Carol is a skilled surgeon and a tennis player. Therefore, Carol is a skilled tennis player.

If we start with the symbolization key we used for the previous argument, we could add a predicate (let Tx mean 'x is a tennis player') and a constant (let c mean Carol). Then the argument becomes:

$$(Rc \,\&\, Kc) \,\&\, Tc$$
$$\therefore\ Tc \,\&\, Kc$$

This translation is a disaster! It takes what in English is a terrible argument and translates it as a valid argument in QL. The problem is that there is a difference between being *skilled as a surgeon* and *skilled as a tennis player.* Translating this argument correctly requires two separate predicates, one for each type of skill. If we let K_1x mean 'x is skilled as a surgeon' and K_2x mean 'x is skilled as a tennis player,' then we can symbolize the argument in this way:

$$(Rc \,\&\, K_1c) \,\&\, Tc$$
$$\therefore\ Tc \,\&\, K_2c$$

Like the English language argument it translates, this is invalid.

The moral of these examples is that you need to be careful of symbolizing predicates in an ambiguous way. Similar problems can arise with predicates like *good*, *bad*, *big*, and *small*. Just as skilled surgeons and skilled tennis players have different skills, big dogs, big mice, and big problems are big in different ways.

Is it enough to have a predicate that means '*x* is a skilled surgeon', rather than two predicates '*x* is skilled' and '*x* is a surgeon'? Sometimes. As sentence 25 shows, sometimes we do not need to distinguish between skilled surgeons and other surgeons.

Must we always distinguish between different ways of being skilled, good, bad, or big? No. As the argument about Billy shows, sometimes we only need to talk about one kind of skill. If you are translating an argument that is just about dogs, it is fine to define a predicate that means '*x* is big.' If the UD includes dogs and mice, however, it is probably best to make the predicate mean '*x* is big for a dog.'

8.13 Multiple quantifiers

Consider this following symbolization key and the sentences that follow it:

> UD: People and dogs
> Dx: *x* is a dog.
> Fxy: *x* is a friend of *y*.
> Oxy: *x* owns *y*.
> f: Fifi
> g: Gerald

26. Fifi is a dog.
27. Gerald is a dog owner.
28. Someone is a dog owner.
29. All of Gerald's friends are dog owners.
30. Every dog owner is the friend of a dog owner.

Sentence 26 is easy: Df.

Sentence 27 can be paraphrased as, 'There is a dog that Gerald owns.' This can be translated as $\exists x(Dx \,\&\, Ogx)$.

Sentence 28 can be paraphrased as, 'There is some *y* such that *y* is a dog owner.' The subsentence '*y* is a dog owner' is just like sentence 27, except that it is

about y rather than being about Gerald. So we can translate sentence 28 as $\exists y \exists x (Dx \,\&\, Oyx)$.

Sentence 29 can be paraphrased as, 'Every friend of Gerald is a dog owner.' Translating part of this sentence, we get $\forall x(Fxg \supset$ 'x is a dog owner'). Again, it is important to recognize that 'x is a dog owner' is structurally just like sentence 27. Since we already have an x-quantifier, we will need a different variable for the existential quantifier. Any other variable will do. Using z, sentence 29 can be translated as $\forall x \big[Fxg \supset \exists z(Dz \,\&\, Oxz) \big]$.

Sentence 30 can be paraphrased as 'For any x that is a dog owner, there is a dog owner who is x's friend.' Partially translated, this becomes

$$\forall x \big[x \text{ is a dog owner} \supset \exists y(y \text{ is a dog owner} \,\&\, Fxy) \big].$$

Completing the translation, sentence 30 becomes

$$\forall x \big[\exists z(Dz \,\&\, Oxz) \supset \exists y \big(\exists z(Dz \,\&\, Oyz) \,\&\, Fxy \big) \big].$$

Consider this symbolization key and these sentences:

> **UD:** people
> **Lxy:** x likes y.
>   **i:** Imre.
>   **k:** Karl.

31. Imre likes everyone that Karl likes.
32. There is someone who likes everyone who likes everyone that he likes.

Sentence 31 can be partially translated as $\forall x(\text{Karl likes } x \supset \text{Imre likes } x)$. This becomes $\forall x(Lkx \supset Lix)$.

Sentence 32 is complex. There is little hope of writing down the whole translation immediately, but we can proceed by small steps. An initial, partial translation might look like this:

$$\exists x \text{ everyone who likes everyone that } x \text{ likes is liked by } x$$

The part that remains in English is a universal sentence, so we translate further:

$$\exists x \forall y(y \text{ likes everyone that } x \text{ likes} \supset x \text{ likes } y).$$

The antecedent of the conditional is structurally just like sentence 31, with y and x in place of Imre and Karl. So sentence 32 can be completely translated in this way

$$\exists x \forall y \big[\forall z(Lxz \supset Lyz) \supset Lxy \big]$$

When symbolizing sentences with multiple quantifiers, it is best to proceed by small steps. Paraphrase the English sentence so that the logical structure is readily symbolized in QL. Then translate piecemeal, replacing the daunting task of translating a long sentence with the simpler task of translating shorter formulae.

8.14 Grammaticality rules for QL

In this section, we provide a formal definition for a *well-formed formula* (wff) and *sentence* of QL.

8.14.1 Expressions

There are six kinds of symbols in QL:

predicates	$A, B, C, \ldots, Z$
with subscripts, as needed	$A_1, B_1, Z_1, A_2, A_{25}, J_{375}, \ldots$
constants	$a, b, c, \ldots, w$
with subscripts, as needed	$a_1, w_4, h_7, m_{32}, \ldots$
variables	x, y, z
with subscripts, as needed	$x_1, y_1, z_1, x_2, \ldots$
sentential connectives	$\neg, \&, \vee, \supset, \equiv$
parentheses	$(\,,\,)$
quantifiers	$\forall, \exists$

We define an EXPRESSION OF QL as any string of symbols of QL. Take any of the symbols of QL and write them down, in any order, and you have an expression.

8.14.2 Well-formed formulae

By definition, a TERM OF QL is either a constant or a variable.

An ATOMIC FORMULA OF QL is an n-place predicate followed by n terms. n here can be any non-negative integer, including 0. (A 0-place predicate is an atomic formula of QL just on its own. It does not require the addition of a term for meaningfulness or a truth value. Note that SL atoms are 0-place QL predicates, and so count as QL atoms too.)

Just as we did for SL, we will give a *recursive* definition for a wff of QL. In fact, most of the definition will look like the definition of a wff of SL: Every

atomic formula is a wff, and you can build new wffs by applying the sentential connectives.

We could just add a rule for each of the quantifiers and be done with it. For instance: If Φ is a wff, then $\forall x \Phi$ and $\exists x \Phi$ are wffs. However, this would allow for some confusing sentences like $\forall x \exists x Dx$. What could these possibly mean? There are possible ways to give interpretations of such sentences, but instead we will write the definition of a wff so that such abominations do not even count as well-formed. QL will include the rule that in order for $\forall x \Phi$ or $\exists x \Phi$ to be a wff, Φ must not already contain an x-quantifier. So $\forall x \exists x Dx$ will not count as a wff because $\exists x Dx$ already contains an x-quantifier.

1. Every atomic formula is a wff.

2. If Φ is a wff, then $\neg \Phi$ is a wff.

3. If Φ and Ψ are wffs, then $(\Phi \mathbin{\&} \Psi)$, is a wff.

4. If Φ and Ψ are wffs, $(\Phi \vee \Psi)$ is a wff.

5. If Φ and Ψ are wffs, then $(\Phi \supset \Psi)$ is a wff.

6. If Φ and Ψ are wffs, then $(\Phi \equiv \Psi)$ is a wff.

7. If Φ is a wff, χ is a variable, and Φ contains no χ-quantifiers, then $\forall \chi \Phi$ is a wff.

8. If Φ is a wff, χ is a variable, and Φ contains no χ-quantifiers, then $\exists \chi \Phi$ is a wff.

9. All and only wffs of QL can be generated by applications of these rules.

Notice that the 'χ' that appears in the definition above is not the variable x. It is a *meta-variable* that stands in for any variable of QL. So $\forall x Ax$ is a wff, but so are $\forall y Ay$, $\forall z Az$, $\forall x_4 Ax_4$, and $\forall z_9 Az_9$.

We can now give a formal definition for scope: The SCOPE of a quantifier is the subformula for which the quantifier is the main logical operator.

8.14.3 Sentences

A sentence is something that can be either true or false. In SL, every wff was a sentence. This will not be the case in QL. Consider the following symbolization key:

   UD: people

Lxy: x loves y
  b: Boris

Consider the expression Lzz. It is an atomic formula: a two-place predicate followed by two terms. All atomic formula are wffs, so Lzz is a wff. Does it mean anything? You might think that it means that z loves himself, in the same way that *Lbb* means that Boris loves himself. But z is a variable; it does not name some person the way a constant would. The wff Lzz does not tell us how to interpret z. Does it mean everyone? Anyone? Someone? Someone in particular? If we had a z-quantifier, it would tell us how to interpret z. For instance, $\exists z Lzz$ would mean that someone loves themself.

Some formal languages treat a wff like Lzz as implicitly having a universal quantifier in front. We will not do this for QL. If you want to say that everyone loves themself, then you need to write the quantifier: $\forall z Lzz$

In order to make sense of a variable, we need a quantifier to tell us how to interpret that variable. The scope of an x-quantifier, for instance, is the part of the formula where the quantifier tells how to interpret x.

In order to be precise about this, we define a BOUND VARIABLE to be an occurrence of a variable χ that is within the scope of an χ-quantifier. A FREE VARIABLE is an occurrence of a variable that is not bound.

For example, consider this wff:

$$\forall x(Ex \lor Dy) \supset \exists z(Rzx \supset Lzx)$$

The scope of the universal quantifier $\forall x$ is $(Ex \lor Dy)$, so the first x is bound by the universal quantifier but the second and third xs are free. There is no y-quantifier, so the y is free. The scope of the existential quantifier $\exists z$ is $(Rzx \supset Lzx)$, so both occurrences of z are bound by it. Since this wff contains variables with no instructions for how to interpret them, we don't know how to evaluate it. It is not a sentence.

We define a SENTENCE of QL as a wff of QL that contains no free variables.

8.14.4 Notational conventions

We will adopt the same notational conventions that we did for SL (p. 40). First, we may leave off the outermost parentheses of a formula. Second, we will sometimes use square brackets '[' and ']' in place of parentheses to increase the readability of formulae. Third, we give ourselves permission to leave out parentheses between each pair of conjuncts when writing long series of conjunctions.

Fourth, we may similarly leave out parentheses between each pair of disjuncts when writing long series of disjunctions.

8.15 Common student errors

A sentence that says everything with one property also has another property should be translated as a universal governing a conditional. Using the obvious interpretation key:

> ▷ 'Every student is working': $\forall x(Sx \supset Wx)$

> ▷ 'Every student has a friend': $\forall x(Sx \supset \exists yFxy)$

> ▷ 'Only students with friends are working': $\forall x[(Sx \,\&\, Wx) \supset \exists yFxy]$

One common error is to translate sentences of this form with a different kind of shape — for example, as a universal governing a conjunction, or as an existential governing a conditional. These are very inaccurate translations of these English sentences:

> ▷ $\forall x(Sx \,\&\, Wx)$

> ▷ $\forall x(Sx \,\&\, \exists yFxy)$

> ▷ $\forall x[(Sx \,\&\, Wx) \,\&\, \exists yFxy]$

These say that *every* object in the UD is a student with the properties in question. Everyone is a student that is working; everyone is a student with a friend; everyone is a working student who has a friend. Any time you have a universal governing a conjunction, you are making a very strong claim — you're not just talking about objects with a particular property, you're saying that multiple things are true about every single object in the domain. Be very careful if you find yourself offering a universal over a conjunction, and make sure you don't mean to use a conditional instead.

It is also a serious mistake to use an existential instead of a universal for sentences like these:

> ▷ $\exists x(Sx \supset Wx)$

> ▷ $\exists x(Sx \supset \exists yFxy)$

> ▷ $\exists x[(Sx \,\&\, Wx) \supset \exists yFxy]$

These are very weak claims. They say that there is some object in the domain that satisfies a certain conditional. For example, $\exists x(Sx \supset Wx)$ says there is something in the domain such that, if it is a student, it is working. Given the truth conditions for the material conditional, this will be true if there is even one object in the domain that is not a student, regardless of who is and isn't working; it will also be true if there is even one object in the domain that is working, regardless of who is and isn't a student.

If you find yourself offering, as a translation of some English sentence, an existential governing a conditional, you are almost certainly making a mistake. This is not a reasonable translation of any ordinary English sentence. You probably want either a universal over a conditional (everything with one property has another property) or an existential over a conjunction (there is something with the following properties).

Practice Exercises

Part A Using the symbolization key given, translate each English-language sentence into QL. Hint: all of these sentences are well-translated as universals governing conditionals.

UD: all humans and animals in the world of *Bojack Horseman*
Dx: x is a dog.
Cx: x is a cat.
Hx: x is a horse.
Bx: x is a human being.
Mx: x is a movie star.
Lxy: x lives with y.
Rxy: x represents y (as y's agent).
Wxy: x worked on a movie with y.
b: Bojack
c: Princess Caroline
d: Dianne
p: Mr. Peanutbutter

1. Every movie star is a dog.
2. Every movie star is a dog or a cat.
3. All dog movie stars live with a human being.
4. Everyone who lives with Dianne is a movie star.
5. Princess Caroline represents every dog movie star.
6. Anyone who worked on a movie with Bojack lives with a movie star.
7. Only humans live with Mr. Peanutbutter.
8. Everyone who has ever worked on a movie with Bojack is either a dog, a cat, or someone who lives with a movie star.

⋆ **Part B** Using the same symbolization key, translate each English-language sentence into QL. Hint: all of these sentences are well-translated as existentials governing conjunctions.

1. Mr. Peanutbutter lives with a human.
2. Dianne lives with a dog who worked on a movie with Bojack.
3. Princess Caroline represents a horse who lives with a human being.
4. Some human being who worked on a movie with Mr. Peanutbutter lives with a dog or a cat.
5. Bojack worked on a movie with a human movie star.
6. Bojack worked on a movie with a nonhuman movie star who lives with Dianne.

Part C Using the same symbolization key, translate each English-language sentence into QL. Hint: these should have different forms than the cases above.

1. If Mr. Peanutbutter is a movie star, then all dogs are movie stars.
2. A dog lives with Dianne and Princess Caroline represents a horse.
3. Princess Caroline represents a horse and a dog.
4. Princess Caroline represents everyone.
5. Dianne doesn't live with anyone.
6. No movie star has ever worked with both Bojack and a cat on a movie.
7. Princess Caroline is a cat, but she doesn't represent any cats.

⋆ **Part D** Using the symbolization key given, translate each English-language sentence into QL.

UD: all animals
 Ax: x is an alligator.
 Mx: x is a monkey.
 Rx: x is a reptile.
 Zx: x lives at the zoo.
 Lxy: x loves y.
  a: Amos
  b: Bouncer
  c: Cleo

1. Amos, Bouncer, and Cleo all live at the zoo.
2. Bouncer is a reptile, but not an alligator.
3. If Cleo loves Bouncer, then Bouncer is a monkey.
4. If both Bouncer and Cleo are alligators, then Amos loves them both.
5. Some reptile lives at the zoo.
6. Every alligator is a reptile.
7. Any animal that lives at the zoo is either a monkey or an alligator.

8. There are reptiles which are not alligators.
9. Cleo loves a reptile.
10. Bouncer loves all the monkeys that live at the zoo.
11. All the monkeys that Amos loves love him back.
12. If any animal is a reptile, then Amos is.
13. If any animal is an alligator, then it is a reptile.
14. Every monkey that Cleo loves is also loved by Amos.
15. There is a monkey that loves Bouncer, but Bouncer does not reciprocate this love.

Part E These are syllogistic figures identified by Aristotle and his successors, along with their medieval names. Translate each argument into QL.

Barbara All Bs are Cs. All As are Bs. ∴. All As are Cs.

Baroco All Cs are Bs. Some A is not B. ∴. Some A is not C.

Bocardo Some B is not C. All As are Bs. ∴. Some A is not C.

Celantes No Bs are Cs. All As are Bs. ∴. No Cs are As.

Celarent No Bs are Cs. All As are Bs. ∴. No As are Cs.

Cemestres No Cs are Bs. No As are Bs. ∴. No As are Cs.

Cesare No Cs are Bs. All As are Bs. ∴. No As are Cs.

Dabitis All Bs are Cs. Some A is B. ∴. Some C is A.

Darii All Bs are Cs. Some A is B. ∴. Some A is C.

Datisi All Bs are Cs. Some A is B. ∴. Some A is C.

Disamis Some B is C. All As are Bs. ∴. Some A is C.

Ferison No Bs are Cs. Some A is B. ∴. Some A is not C.

Ferio No Bs are Cs. Some A is B. ∴. Some A is not C.

Festino No Cs are Bs. Some A is B. ∴. Some A is not C.

Baralipton All Bs are Cs. All As are Bs. ∴. Some C is A.

Frisesomorum Some B is C. No As are Bs. ∴. Some C is not A.

⋆ **Part F** Using the symbolization key given, translate each English-language sentence into QL.

UD: all animals

> **Dx:** x is a dog.
> **Sx:** x likes samurai movies.
> **Lxy:** x is larger than y.
> **b:** Bertie
> **e:** Emerson
> **f:** Fergis

1. Bertie is a dog who likes samurai movies.
2. Bertie, Emerson, and Fergis are all dogs.
3. Emerson is larger than Bertie, and Fergis is larger than Emerson.
4. All dogs like samurai movies.
5. Only dogs like samurai movies.
6. There is a dog that is larger than Emerson.
7. If there is a dog larger than Fergis, then there is a dog larger than Emerson.
8. No animal that likes samurai movies is larger than Emerson.
9. No dog is larger than Fergis.
10. Any animal that dislikes samurai movies is larger than Bertie.
11. There is an animal that is between Bertie and Emerson in size.
12. There is no dog that is between Bertie and Emerson in size.
13. No dog is larger than itself.
14. For every dog, there is some dog larger than it.
15. There is an animal that is smaller than every dog.

Part G For each argument, write a symbolization key and translate the argument into QL.

1. Nothing on my desk escapes my attention. There is a computer on my desk. As such, there is a computer that does not escape my attention.
2. All my dreams are black and white. Old TV shows are in black and white. Therefore, some of my dreams are old TV shows.
3. Neither Holmes nor Watson has been to Australia. A person could see a kangaroo only if they had been to Australia or to a zoo. Although Watson has not seen a kangaroo, Holmes has. Therefore, Holmes has been to a zoo.
4. No one expects the Spanish Inquisition. No one knows the troubles I've seen. Therefore, anyone who expects the Spanish Inquisition knows the troubles I've seen.
5. An antelope is bigger than a bread box. I am thinking of something that is no bigger than a bread box, and it is either an antelope or a cantaloupe. As such, I am thinking of a cantaloupe.
6. All babies are illogical. Nobody who is illogical can manage a crocodile. Berthold is a baby. Therefore, Berthold is unable to manage a crocodile.

⋆ **Part H** Using the symbolization key given, translate each English-language sentence into QL.

UD: candies
Cx: x has chocolate in it.
Mx: x has marzipan in it.
Sx: x has sugar in it.
Tx: Boris has tried x.
Bxy: x is better than y.

1. Boris has never tried any candy.
2. Marzipan is always made with sugar.
3. Some candy is sugar-free.
4. The very best candy is chocolate.
5. No candy is better than itself.
6. Boris has never tried sugar-free chocolate.
7. Boris has tried marzipan and chocolate, but never together.
8. Any candy with chocolate is better than any candy without it.
9. Any candy with chocolate and marzipan is better than any candy that lacks both.

⋆ **Part I** Using the symbolization key given, translate each English-language sentence into QL.

UD: people and dishes at a potluck
Rx: x has run out.
Tx: x is on the table.
Fx: x is food.
Px: x is a person.
Lxy: x likes y.
e: Eli
f: Francesca
g: the guacamole

1. All the food is on the table.
2. If the guacamole has not run out, then it is on the table.
3. Everyone likes the guacamole.
4. If anyone likes the guacamole, then Eli does.
5. Francesca only likes the dishes that have run out.
6. Francesca likes no one, and no one likes Francesca.
7. Eli likes anyone who likes the guacamole.
8. Eli likes everyone who likes anyone that he likes.
9. If there is a person on the table already, then all of the food must have run out.

⋆ **Part J** Using the symbolization key given, translate each English-language sentence into QL.

UD: people
Dx: x dances ballet.
Fx: x is female.
Mx: x is male.
Cxy: x is a child of y.
Sxy: x is a sibling of y.
e: Elmer
j: Jane
p: Patrick

1. All of Patrick's children are ballet dancers.
2. Jane is Patrick's daughter.
3. Patrick has a daughter.
4. Jane is an only child.
5. All of Patrick's daughters dance ballet.
6. Patrick has no sons.
7. Jane is Elmer's niece.
8. Patrick is Elmer's brother.
9. Patrick's brothers have no children.
10. Jane is an aunt.
11. Everyone who dances ballet has a sister who also dances ballet.
12. Every man who dances ballet is the child of someone who dances ballet.

Part K Identify which variables are bound and which are free.

1. $\exists x Lxy \mathbin{\&} \forall y Lyx$
2. $\forall x Ax \mathbin{\&} Bx$
3. $\forall x (Ax \mathbin{\&} Bx) \mathbin{\&} \forall y (Cx \mathbin{\&} Dy)$
4. $\forall x \exists y [Rxy \supset (Jz \mathbin{\&} Kx)] \lor Ryx$
5. $\forall x_1 (Mx_2 \equiv Lx_2 x_1) \mathbin{\&} \exists x_2 Lx_3 x_2$

⋆ **Part L**

1. Identify which of the following are substitution instances of $\forall x Rcx$: *Rac*, *Rca*, *Raa*, *Rcb*, *Rbc*, *Rcc*, *Rcd*, *Rcx*
2. Identify which of the following are substitution instances of $\exists x \forall y Lxy$: $\forall y Lby$, $\forall x Lbx$, *Lab*, $\exists x Lxa$

Chapter 9

A formal semantics for QL

In this chapter, we describe a *formal semantics* for QL. This corresponds to the discussion of interpretations and truth in SL given in Chapter 4. Like truth in SL, truth in QL is defined relative to a particular interpretation; entailment is a matter of truth in all interpretations. In SL, models corresponded to assignments of truth values to atomic sentences. For example, here is an SL model:

$$I = \left\{ \begin{array}{l} P = 0 \\ Q = 1 \\ R = 0 \end{array} \right.$$

Interpretation I settles the truth value of any SL sentence one can construct from P, Q, and R. Because QL involves richer notions and a more complex vocabulary than SL, it requires correspondingly richer models and interpretations.

9.1 Interpretations in QL

What is an interpretation in QL? Like a symbolization key for QL, an interpretation requires a universe of discourse, a schematic meaning for each of the predicates, and an object that is picked out by each constant. For example:

UD: Marvel characters
Hx: x is a hero.
Sx: x has spider powers.
m: Miles Morales
p: Peter Parker

r: The Red Skull
s: Susan Storm
u: Ultimate Spider-Man

This interpretation is given in terms of English descriptions. To apply it, you need to know some details about the characters in question. For example, Sm is true on this interpretation, because Miles Morales does have spider powers. But the interpretation itself doesn't tell us that — to get that information from this way of setting out the interpretation, you need to know some details about what happens in the story. You need to know, for example, that Miles Morales, like his more famous mentor Peter Parker, also has spider powers. If you do know a bit about Marvel comics, you may know that Miles Morales is actually the Ultimate Spider-Man. So u and m in this interpretation are two different names for the same member of the UD. There is no rule against having multiple names for the same member. (We'll discuss this issue in much more detail in Chapter 12.)

We want our QL models to encode this kind of information too. Like a good SL model, a QL model shouldn't require prior knowledge of comic books. One way we could try to do this would be to just give a truth value assignment, as we did for SL. The truth value assignment would assign 0 or 1 to each atomic wff: $Sm = 1$, $Sp = 1$, $Sr = 0$, and so on. If we were to do that, however, we might just as well translate the sentences from QL to SL by replacing Sp and Sm with sentence letters. We could then rely on the definition of truth for SL, but at the cost of ignoring all the logical structure of predicates and terms. In writing a symbolization key for QL, we do not give separate definitions for Sp and Sm. Instead, we give meanings to the components S, p, and m. This is essential because we want to be able to reflect the logical relationships between e.g. Sp and $\exists x Sx$.

Our interpretations should include explanations for predicates and names, not just for sentences. We cannot use a truth value assignment for this, because a predicate by itself is neither true nor false. In the interpretation given above, H is true *of* Peter Parker (i.e., Hp is true), but it makes no sense at all to ask whether H on its own is true. It would be like asking whether the English language fragment '. . .is a hero' is true.

What does an interpretation do for a predicate, if it does not make it true or false? An interpretation helps to pick out the objects to which the predicate applies. Interpreting Hx to mean 'x is a hero' picks out some characters as the things that are Hs. Formally, this is a set of members of the UD to which the predicate applies; this set is called the EXTENSION of the predicate.

Some predicates have indefinitely large extensions. It would be impractical to try and write down all of the Marvel characters individually, so instead we use an English language expression to interpret the predicate. This is somewhat

imprecise, because the interpretation alone does not tell you which members of the UD are in the extension of the predicate. In order to figure out whether a particular member of the UD is in the extension of the predicate (to figure out whether the Red Skull is a hero, for instance), you need to know about comic books. (As you might guess from his name, he's not.) In general, the extension of a predicate is the result of an interpretation *along with* some facts.

Sometimes it is possible to list all of the things that are in the extension of a predicate. Instead of writing a schematic English sentence, we can write down the extension as a set of things. Suppose we wanted to add a one-place predicate F to the key above, meaning 'x is a founding member of the Fantastic Four', so we write the extension as a set of characters:

extension(F) = {Reed Richards, Susan Storm, Johnny Storm, Ben Grimm}

You do not need to know anything about comic books to be able to determine that, on this interpretation, Fs is true: Susan Storm, whose name is given as s, is just specified to be one of the things that is F. Similarly, $\exists x F x$ is obviously true on this interpretation: There is at least one member of the UD that is an F — in fact, there are four of them.

What about the sentence $\forall x F x$? The sentence is false, because it is not true that all members of the UD are F. It requires the barest minimum of knowledge about comic books to know that there are other characters besides just these four. Although we specified the extension of F in a formally precise way, we still specified the UD with an English language description. Formally speaking, a UD is just a set of members.

The formal significance of a predicate is determined by its extension, but what should we say about constants like m and s? The meaning of a constant determines which member of the UD is picked out by the constant. The individual that the constant picks out is called the REFERENT of the constant. Both m and u have the same referent, since they both refer to the same comic book character. You can think of a constant letter as a name and the referent as the thing named. In English, we can use the different names 'Miles' and 'Ultimate Spider-Man' to refer to the same comic book character. In this interpretation, we also use the different constants 'm' and 'u' to refer to the same member of the UD.

9.2 Sets

We use curly brackets '{' and '}' to denote sets. The members of the set can be listed in any order, separated by commas. This means that {foo, bar} and {bar, foo} are the same set.

It is possible to have a set with no members in it. This is called the EMPTY SET. The empty set is sometimes written as {}, but usually it is written as the single symbol ∅.

9.3 Models

As we have seen, an interpretation in QL is only formally significant insofar as it determines a UD, an extension for each predicate, and a referent for each constant. We call this formal structure a MODEL for QL.

To see how this works, consider this symbolization key:

UD: The first ten natural numbers
Px: x is prime.
n_4: 4

Given some basic mathematical knowledge, it is obvious that Pn_4 is false. Let's consider the model this interpretation suggests, to show why it makes this wff false. Instead of just giving a description in the UD, we can list the members as a set. We also define the extension of the predicate P, and the referent of the constant n_4:

$$UD = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$
$$extension(P) = \{2, 3, 5, 7\}$$
$$referent(n_4) = 4$$

This is not a full model for this interpretation, but it is a detailed enough partial model to show that Pn_4 is false. You do not need to know anything about mathematics to see that this sentence is false in this model. The UD member named by n_4 is not in the extension of P. In this way, the model captures all of the formal significance of the interpretation.

Suppose we enrich this interpretation with more predicates:

UD: The first ten natural numbers
Ex: x is even.
Nx: x is negative.
Lxy: x is less than y.
Txyz: x times y equals z.

What do we need to add to the model for our new predicates?

The extension of E is the subset $\{2, 4, 6, 8, 10\}$. There are no negative numbers in the UD, so N has an empty extension; i.e. $extension(N) = \varnothing$.

Sometimes it will be convenient to represent extensions graphically, similar to the way we did with truth tables. We can represent the extensions just described for P, E, and N thus:

	P	E	N
1	0	0	0
2	1	1	0
3	1	0	0
4	0	1	0
5	1	0	0
6	0	1	0
7	1	0	0
8	0	1	0
9	0	0	0
10	0	1	0

The members of the UD are listed as rows; the predicates are given as columns. The 0s and 1s indicate whether each member satisfies each predicate. Notice that the same information is conveyed in this chart as in the three sets of ordered pairs described above. Either is an acceptable way of indicating the extension of the predicates.

The extension of a two-place predicate like L is more complicated. No individual number falls under the extension of this predicate; it is about the relation between members. Note also that sets of pairs numbers aren't suitable for the extension of L either, because 1 is less than 8, but 8 is not less than 1. And the set {1,8} is the very same set as the set {8,1}. The solution is to have the extension of L consist in a set of ORDERED PAIRS of numbers. An ordered pair is like a set with two members, except that the order *does* matter. We write ordered pairs with angle brackets '<' and '>'. The ordered pair <foo, bar> is different than the ordered pair <bar, foo>. The extension of L is a collection of ordered pairs, all of the pairs of numbers in the UD such that the first number is less than the second. Writing this out completely:

extension(L) = {<1, 2>, <1, 3>, <1, 4>, <1, 5>, <1, 6>, <1, 7>, <1, 8>, <1, 9>, <1, 10>, <2, 3>, <2, 4>, <2, 5>, <2, 6>, <2, 7>, <2, 8>, <2, 9>, <2, 10>, <3, 4>, <3, 5>, <3, 6>, <3, 7>, <3, 8>, <3, 9>, <3, 10>, <4, 5>, <4, 6>, <4, 7>, <4, 8>, <4, 9>, <4, 10>, <5, 6>, <5, 7>, <5, 8>, <5, 9>, <5, 10>, <6, 7>, <6, 8>, <6, 9>, <6, 10>, <7, 8>, <7, 9>, <7, 10>, <8, 9>, <8, 10> <9, 10>}

Formally, the extension of a two-place predicate is a set of ordered pairs. Sometimes we will find it easier to represent extensions in a chart, with the two variable positions represented on the two axes. For example, the extension above could be expressed via a table like the one below. The '0' in the first cell of the table says that <1, 1> is not in the extension of L; the next cell in the first row

says that <1, 2> is. Etc. Sometimes drawing out a chart like this will be the easiest way to represent models for the extensions of two-place predicates.

Lxy	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	1	1	1	1	1	1
2	0	0	1	1	1	1	1	1	1	1
3	0	0	0	1	1	1	1	1	1	1
4	0	0	0	0	1	1	1	1	1	1
5	0	0	0	0	0	1	1	1	1	1
6	0	0	0	0	0	0	1	1	1	1
7	0	0	0	0	0	0	0	1	1	1
8	0	0	0	0	0	0	0	0	1	1
9	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0

The extension of a three-place predicate is a set of ordered triples where the predicate is true of those three things *in that order*. So the extension of T in this model will contain ordered triples like <2, 4, 8>, because $2 \times 4 = 8$. Because the surface of a sheet of paper is for all intents and purposes two-dimensional, it is usually not convenient to represent 3-or-more place predicates with tables.

Generally, the extension of an n-place predicate is a set of all ordered n-tuples $\langle a_1, a_2, \ldots, a_n \rangle$ such that a_1–a_n are members of the UD and the predicate is true of a_1–a_n in that order.

9.4 Working with models

We will use the double turnstile symbol for QL much as we did for SL. '$\Phi \models \Psi$' means that 'Φ entails Ψ': When Φ and Ψ are two sentences of QL, $\Phi \models \Psi$ means that there is no model in which Φ is true and Ψ is false. $\models \Phi$ is shorthand for $\varnothing \models \Phi$, which means that Φ is true in every model. This allows us to give definitions for various concepts in QL. In fact, we can use the same definitions offered in Chapter 4.

> A TAUTOLOGY IN QL is a sentence Φ that is true in every model; i.e., $\models \Phi$.
>
> A CONTRADICTION IN QL is a sentence Φ that is false in every model; i.e., $\models \neg\Phi$.
>
> A sentence is CONTINGENT IN QL if and only if it is neither a tautology nor a contradiction.
>
> An argument " $\mathcal{P}_1, \mathcal{P}_2, \cdots, \therefore \Omega$ " is VALID IN QL if and only if there is no model in which all of the premises are true and the conclusion is false; i.e., $\{\mathcal{P}_1, \mathcal{P}_2, \cdots\} \models \Omega$. It is INVALID IN QL otherwise.

Two sentences Φ and Ψ are LOGICALLY EQUIVALENT IN QL if and only if both $\Phi \models \Psi$ and $\Psi \models \Phi$.

The set $\{\Phi_1, \Phi_2, \Phi_3, \cdots\}$ is CONSISTENT IN QL if and only if there is at least one model in which all of the sentences are true. The set is INCONSISTENT IN QL if and only if there is no such model.

9.5 Constructing models

Suppose we want to show that $\forall x Axx \supset Bd$ is *not* a tautology. This requires showing that the sentence is not true in every model. If we can provide an example of a model in which the sentence is false, then we will have shown that the sentence is not a tautology.

What would such a model look like? In order for $\forall x Axx \supset Bd$ to be false, the antecedent ($\forall x Axx$) must be true, and the consequent (Bd) must be false.

To construct such a model, we start with a UD. It will be easier to specify extensions for predicates if we have a small UD, so start with a UD that has just one member. Formally, this single member might be anything. Let's say it is Miles Morales.

We want $\forall x Axx$ to be true, so we want all members of the UD to be paired with themself in the extension of A; this means that the extension of A must be {<Miles Morales,Miles Morales>}.

We want Bd to be false, so the referent of d must not be in the extension of B. We give B an empty extension.

Since Miles is the only member of the UD, it must be the referent of d. The model we have constructed looks like this:

$$\text{UD} = \{\text{Miles Morales}\}$$
$$\text{extension}(A) = \{<\text{Miles Morales, Miles Morales}>\}$$
$$\text{extension}(B) = \varnothing$$
$$\text{referent}(d) = \text{Miles Morales}$$

Strictly speaking, a model specifies an extension for *every* predicate of QL and a referent for *every* constant. As such, it is generally impossible to write down a complete model. That would require writing down infinitely many extensions and infinitely many referents. However, we do not need to consider every predicate in order to show that there are models in which $\forall x Axx \supset Bd$ is false. Predicates like H and constants like f_{13} make no difference to the truth or falsity of this sentence. It is enough to specify extensions for A and B and a referent for d, as we have done. This provides a *partial model* in which the sentence is false.

Perhaps you are wondering: What does the predicate A mean in English? It doesn't really matter. For formal purposes, the existence of models like the ones described above are enough to show that $\forall x A x x \supset B d$ is not a tautology. But we can offer an interpretation in English if we like. How about this one?

> **UD:** Miles Morales
> Axy**:** x knows y's biggest secret.
> Bx**:** x's powers derive from gamma radiation.
> d**:** Miles Morales

This is one way we can interpret the model above. *Add* is true, because Miles does know Miles's biggest secret. (It's that he's the Ultimate Spider-Man. Now you know it too!) *Bd* is false: Miles's powers came from a genetically enhanced spider bite, not from gamma radiation. But the partial model constructed above includes none of these interpretative details. All it says is that A is a predicate which is true of Miles and Miles, and that B is a predicate which does not apply to Miles. There are indefinitely many predicates in English that have this extension. Axy might instead translate 'x is the same size as y' or 'x and y live in the same city'; Bx might translate 'x is a billionaire' or 'x's uncle was killed by a robber' or 'Donald Trump has written a tweet about x'. In constructing a model and giving extensions for A and B, we do not specify what English predicates A and B should be used to translate. We are concerned with whether the $\forall x A x x \supset B d$ comes out true or false, and all that matters for truth and falsity in QL is the information in the model: the UD, the extensions of predicates, and the referents of constants.

We can just as easily show that $\forall x A x x \supset B d$ is not a contradiction. We need only specify a model in which $\forall x A x x \supset B d$ is true; i.e., a model in which either $\forall x A x x$ is false or $B d$ is true. Here is one such partial model:

$$\text{UD} = \{\text{The Red Skull}\}$$
$$\text{extension}(A) = \{<\text{The Red Skull, The Red Skull}>\}$$
$$\text{extension}(B) = \{\text{The Red Skull}\}$$
$$\text{referent}(d) = \text{The Red Skull}$$

I've switched our object from Miles Morales to The Red Skull to emphasize that it doesn't matter what object you pick. (Changing the examples all back to Miles would make no difference.) On this model, $\forall x A x x \supset B d$ is true, because it is a conditional with a true consequent (as well as a true antecedent). We have now shown that $\forall x A x x \supset B d$ is neither a tautology nor a contradiction. By the definition of 'contingent in QL,' this means that $\forall x A x x \supset B d$ is contingent. In general, showing that a sentence is contingent will require two models: one in which the sentence is true and another in which the sentence is false.

Suppose we want to show that $\forall x S x$ and $\exists x S x$ are not logically equivalent. We need to construct a model in which the two sentences have different truth values;

we want one of them to be true and the other to be false. We start by specifying a UD. Again, we make the UD reasonably small so that we can specify extensions easily. But this time we will need at least two members. If we only had one member of the domain, we wouldn't be able to illustrate the difference between *all* and *some*. Let's let our UD be {The Red Skull, Miles Morales}.

We can make $\exists x Sx$ true by including something in the extension of S, and we can make $\forall x Sx$ false by leaving something out of the extension of S. It does not matter which one we include and which one we leave out. Making Miles the only S, we get a partial model that looks like this:

$$UD = \{\text{Miles, The Red Skull}\}$$
$$\text{extension}(S) = \{\text{Miles}\}$$

This partial model shows that the two sentences are *not* logically equivalent. $\exists x Sx$ is assigned 1 on this model, but $\forall x Sx$ is assigned 0.

Back on p. 149, we said that this argument would be invalid in QL:

$$(Rc \,\&\, K_1 c) \,\&\, Tc$$
$$\therefore \; Tc \,\&\, K_2 c$$

Now we can prove that this is so. To show that this argument is invalid, we need to show that there is some model in which the premise is true and the conclusion is false. We can construct such a model deliberately. Here is one way to do it:

$$UD = \{\text{Reed Richards}\}$$
$$\text{extension}(T) = \{\text{Reed Richards}\}$$
$$\text{extension}(K_1) = \{\text{Reed Richards}\}$$
$$\text{extension}(K_2) = \varnothing$$
$$\text{extension}(R) = \{\text{Reed Richards}\}$$
$$\text{referent}(c) = \text{Reed Richards}$$

Similarly, we can show that a set of sentences is consistent by constructing a model in which all of the sentences are true.

9.6 Reasoning about all models

We can show that a sentence is *not* a tautology just by providing one carefully specified model: a model in which the sentence is false. To show that something is a tautology, on the other hand, it would not be enough to construct ten, one hundred, or even a thousand models in which the sentence is true. It is only a tautology if it is true in *every* model, and there are infinitely many models.

Table 9.1: It is relatively easy to answer a question if you can do it by constructing a model or two. It is much harder if you need to reason about all possible models. This table shows when constructing models is enough.

	YES	NO
Is Φ a tautology?	show that Φ must be true in any model	*construct a model* in which Φ is false
Is Φ a contradiction?	show that Φ must be false in any model	*construct a model* in which Φ is true
Is Φ contingent?	*construct two models*, one in which Φ is true and another in which Φ is false	either show that Φ is a tautology or show that Φ is a contradiction
Are Φ and Ψ equivalent?	show that Φ and Ψ must have the same truth value in any model	*construct a model* in which Φ and Ψ have different truth values
Is the set $\mathbb{A}$ consistent?	*construct a model* in which all the sentences in $\mathbb{A}$ are true	show that the sentences in $\mathbb{A}$ could not all be true in any model
Is the argument '$\mathcal{P}$, $\therefore$ Ω' valid?	show that any model in which $\mathcal{P}$ is true must be a model in which Ω is true	*construct a model* in which $\mathcal{P}$ is true and Ω is false

This cannot be avoided just by constructing partial models, because there are infinitely many partial models.

Consider, for example, the sentence $Raa \equiv Raa$. There are two logically distinct partial models of this sentence that have a 1-member UD. There are 32 distinct partial models that have a 2-member UD. There are 1526 distinct partial models that have a 3-member UD. There are 262,144 distinct partial models that have a 4-member UD. And so on to infinity. In order to show that this sentence is a tautology, we need to show something about all of these models. There is no hope of doing so by dealing with them one at a time.

Nevertheless, $Raa \equiv Raa$ is obviously a tautology. We can prove it with a simple argument:

> There are two kinds of models: those with $\langle \text{referent(a)}, \text{referent(a)} \rangle$ is in the extension of R and those in which it is not. In the first kind of model, Raa is true; by the truth table for the biconditional, $Raa \equiv Raa$ is also true. In the second kind of model, Raa is false;

> this makes $Raa \equiv Raa$ true. Since the sentence is true in both kinds of model, and since every model is one of the two kinds, $Raa \equiv Raa$ is true in every model. Therefore, it is a tautology.

This is a sound argument; it should convince us of its conclusion. But note that it is not an argument in QL. Rather, it is an argument in English *about* QL; it is an argument in the metalanguage. There is no formal procedure for evaluating or constructing natural language arguments like this one. The imprecision of natural language is the very reason we began thinking about formal languages.

There are further difficulties with this approach.

Consider the sentence $\forall x(Rxx \supset Rxx)$, another obvious tautology. It might be tempting to reason in this way: '$Rxx \supset Rxx$ is true in every model, so $\forall x(Rxx \supset Rxx)$ must be true.' The problem is that $Rxx \supset Rxx$ is *not* true in every model. It is not a sentence, and so it is *neither* true *nor* false. We do not yet have the vocabulary to say what we want to say about $Rxx \supset Rxx$. In the next section, we introduce the concept of *satisfaction*; after doing so, we will be better able to provide an argument that $\forall x(Rxx \supset Rxx)$ is a tautology.

It is necessary to reason about an infinity of models to show that a sentence is a tautology. Similarly, it is necessary to reason about an infinity of models to show that a sentence is a contradiction, that two sentences are equivalent, that a set of sentences is inconsistent, or that an argument is valid. There are other things we can show by carefully constructing a model or two. Table 9.1 summarizes which things are which.

9.7 Truth in QL

For SL, we split the definition of truth into two parts: a truth value assignment (a) for sentence letters and a truth function (v) for all sentences. The truth function covered the way that complex sentences could be built out of sentence letters and connectives.

In the same way that truth for SL is always *truth given a truth value assignment*, truth for QL is *truth in a model*. The atomic sentences, again, are n-place predicates followed by n constants, like Pj. It is true in a model $\mathbb{M}$ if and only if the referent of j is in the extension of P in $\mathbb{M}$.

We could go on in this way to define truth for all atomic sentences that contain only predicates and constants: Consider any sentence of the form $\mathcal{R}a_1 \ldots a_n$ where $\mathcal{R}$ is an n-place predicate and the as are constants. It is true in $\mathbb{M}$ if and only if $\langle \text{referent}(a_1), \ldots, \text{referent}(a_n) \rangle$ is in extension($\mathcal{R}$) in $\mathbb{M}$.

We could then define truth for sentences built up with sentential connectives in the same way we did for SL. For example, the sentence $(Pj \supset Mda)$ is true in $\mathbb{M}$ if either Pj is false in $\mathbb{M}$ or Mda is true in $\mathbb{M}$.

Unfortunately, this approach will fail when we consider sentences containing quantifiers. Consider $\forall x Px$. When is it true in a model $\mathbb{M}$? The answer cannot depend on whether Px is true or false in $\mathbb{M}$, because the x in Px is a free variable. Px is not a sentence. It is neither true nor false.

We were able to give a recursive definition of truth for SL because every well-formed formula of SL has a truth value. This is not true in QL, so we cannot define truth by starting with the truth of atomic sentences and building up. We also need to consider the atomic formulae which are not sentences. In order to do this we will define *satisfaction*; every well-formed formula of QL will be satisfied or not satisfied, even if it does not have a truth value. We will then be able to define *truth* for sentences of QL in terms of satisfaction.

9.8 Satisfaction

The formula Px says, roughly, that x is one of the Ps. This cannot be quite right, however, because x is a variable and not a constant. It does not name any particular member of the UD. Instead, its meaning in a sentence is determined by the quantifier that binds it. The variable x must stand-in for every member of the UD in the sentence $\forall x Px$, but it only needs to stand-in for one member in $\exists x Px$. Since we want the definition of satisfaction to cover Px without any quantifier whatsoever, we will start by saying how to interpret a free variable like the x in Px.

We do this by introducing a *variable assignment*. Formally, this is a function that matches up each variable with a member of the UD. Call this function 'a'. (The 'a' is for 'assignment', but this is not the same as the truth value assignment that we used in defining truth for SL.)

The formula Px is satisfied in a model $\mathbb{M}$ by a variable assignment a if and only if $a(x)$, the object that a assigns to x, is in the extension of P in $\mathbb{M}$.

When is $\forall x Px$ satisfied? It is not enough if Px is satisfied in $\mathbb{M}$ by a, because that just means that $a(x)$ is in extension(P). $\forall x Px$ requires that every other member of the UD be in extension(P) as well.

So we need another bit of technical notation: For any member π of the UD and any variable χ, let $a[\pi|\chi]$ be the variable assignment that assigns π to χ but agrees with a in all other respects. We have used π, the Greek letter *pi*, to underscore the fact that it is some member of the UD and not some symbol

of QL. Suppose, for example, that the UD is presidents of the United States. The function $a[\text{Grover Cleveland}|x]$ assigns Grover Cleveland to the variable x, regardless of what a assigns to x; for any other variable, $a[\text{Grover Cleveland}|x]$ agrees with a.

We can now say concisely that $\forall x Px$ is satisfied in a model $\mathbb{M}$ by a variable assignment a if and only if, for every object π in the UD of $\mathbb{M}$, Px is satisfied in $\mathbb{M}$ by $a[\pi|x]$.

The intuitive thought here is that wff satisfaction is relative to a variable assignment. A variable assignment is a way of treating each variable as if it were a name for some object or other; a wff is satisfied by a in a given model iff, in that model, treating the variables the way a suggests would yield a true wff.

You may worry that our statement of satisfaction by a variable assignment in a model is circular, because it gives the satisfaction conditions for the sentence $\forall x Px$ using the phrase 'for every object.' However, it is important to remember the difference between a logical symbol like '$\forall$' and an English language word like 'every.' The word is part of the metalanguage that we use in defining satisfaction conditions for object language sentences that contain the symbol.

We can now give a general definition of satisfaction, extending from the cases we have already discussed. We define a function s (for 'satisfaction') in a model $\mathbb{M}$ such that for any wff Φ and variable assignment a, $s(\Phi, a) = 1$ if Φ is satisfied in $\mathbb{M}$ by a; otherwise $s(\Phi, a) = 0$.

1. If Φ is an atomic wff of the form $\mathcal{P}t_1 \dots t_n$ and π_i is the object picked out by t_i, then

$$s(\Phi, a) = \left\{ \begin{array}{ll} 1 & \text{if } \langle \pi_1 \dots \pi_n \rangle \text{ is in extension}(\mathcal{P}) \text{ in } \mathbb{M}, \\ 0 & \text{otherwise.} \end{array} \right.$$

   For each term t_i: If t_i is a constant, then $\pi_i = \text{referent}(t_i)$. If t_i is a variable, then $\pi_i = a(t_i)$.

2. If Φ is $\neg \Psi$ for some wff Ψ, then

$$s(\Phi, a) = \left\{ \begin{array}{ll} 1 & \text{if } s(\Psi, a) = 0, \\ 0 & \text{otherwise.} \end{array} \right.$$

3. If Φ is $(\Psi \,\&\, \Omega)$ for some wffs Φ, Ψ, then

$$s(\Phi, a) = \left\{ \begin{array}{ll} 1 & \text{if } s(\Psi, a) = 1 \text{ and } s(\Omega, a) = 1, \\ 0 & \text{otherwise.} \end{array} \right.$$

4. If Φ is $(\Psi \lor \Omega)$ for some wffs Φ, Ψ, then

$$s(\Phi, a) = \left\{ \begin{array}{ll} 0 & \text{if } s(\Psi, a) = 0 \text{ and } s(\Omega, a) = 0, \\ 1 & \text{otherwise.} \end{array} \right.$$

5. If Φ is $(\Psi \supset \Omega)$ for some wffs Φ, Ψ, then

$$s(\Phi, a) = \begin{cases} 0 & \text{if } s(\Psi, a) = 1 \text{ and } s(\Omega, a) = 0, \\ 1 & \text{otherwise.} \end{cases}$$

6. If Φ is $(\Psi \equiv \Omega)$ for some wffs Φ, Ψ, then

$$s(\Phi, a) = \begin{cases} 1 & \text{if } s(\Psi, a) = s(\Omega, a), \\ 0 & \text{otherwise.} \end{cases}$$

7. If Φ is $\forall \chi \Psi$ for some wff Ψ and some variable χ, then

$$s(\Phi, a) = \begin{cases} 1 & \text{if } s(\Psi, a[\pi|\chi]) = 1 \text{ for every member } \pi \text{ of the UD}, \\ 0 & \text{otherwise.} \end{cases}$$

8. If Φ is $\exists \chi \Psi$ for some wff Ψ and some variable χ, then

$$s(\Phi, a) = \begin{cases} 1 & \text{if } s(\Psi, a[\pi|\chi]) = 1 \text{ for at least one member } \pi \text{ of the UD}, \\ 0 & \text{otherwise.} \end{cases}$$

This definition follows the same structure as the definition of a wff for QL, so we know that every wff of QL will be covered by this definition. For a model $\mathbb{M}$ and a variable assignment a, any wff will either be satisfied or not. No wffs are left out or assigned conflicting values.

9.9 Truth in QL

Consider a simple sentence like $\forall x Px$. By part 7 in the definition of satisfaction, this sentence is satisfied if $a[\pi|x]$ satisfies Px in $\mathbb{M}$ for every π in the UD. In other words, assign that x to any object in the UD you like, and the resultant wff will come out true. By part 1 of the definition, this will be the case if every π is in the extension of P. Whether $\forall x Px$ is satisfied does not depend on the particular variable assignment a. If this sentence is satisfied, then it is true. This is a formalization of what we have said all along: $\forall x Px$ is true if everything in the UD is in the extension of P.

The same thing holds for any sentence of QL. Because all of the variables are bound, a sentence is satisfied or not regardless of the details of the variable assignment. So we can define truth in this way: A sentence Φ is TRUE IN $\mathbb{M}$ if and only if some variable assignment satisfies Φ in M; Φ is FALSE IN $\mathbb{M}$ otherwise.

Truth in QL is *truth in a model*. Sentences of QL are not flat-footedly true or false as mere symbols, but only relative to a model. A model provides the meaning of the symbols, insofar as it makes any difference to truth and falsity.

9.10 Reasoning about all models (reprise)

At the end of section 9.6, we were stymied when we tried to show that $\forall x(Rxx \supset Rxx)$ is a tautology. Having defined satisfaction, we can now reason in this way:

> Consider some arbitrary model $\mathbb{M}$. Now consider an arbitrary member of the UD; for the sake of convenience, call it π. It must be the case either that $\langle \pi, \pi \rangle$ is in the extension of R or that it is not. If $\langle \pi, \pi \rangle$ is in the extension of R, then Rxx is satisfied by a variable assignment that assigns π to x (by part 1 of the definition of satisfaction); since the consequent of $Rxx \supset Rxx$ is satisfied, the conditional is satisfied (by part 5). If $\langle \pi, \pi \rangle$ is not in the extension of R, then Rxx is not satisfied by a variable assignment that assigns π to x (by part 1); since antecedent of $Rxx \supset Rxx$ is not satisfied, the conditional is satisfied (by part 5). In either case, $Rxx \supset Rxx$ is satisfied. This is true for any member of the UD, so $\forall x(Rxx \supset Rxx)$ is satisfied by any truth value assignment (by part 7). So $\forall x(Rxx \supset Rxx)$ is true in $\mathbb{M}$ (by the definition of truth). This argument holds regardless of the exact UD and regardless of the exact extension of R, so $\forall x(Rxx \supset Rxx)$ is true in any model. Therefore, it is a tautology.

Giving arguments about all possible models typically requires clever combination of two strategies:

1. Divide cases between two possible kinds, such that every case must be one kind or the other. In the argument on p. 171, for example, we distinguished two kinds of models based on whether or not a specific ordered pair was in extension(R). In the argument above, we distinguished cases in which an ordered pair was in extension(R) and cases in which it was not.

2. Consider an arbitrary object as a way of showing something more general. In the argument above, it was crucial that π was just some arbitrary member of the UD. We did not assume anything special about it. As such, whatever we could show to hold of π must hold of every member of the UD — if we could show it for π, we could show it for anything. In the same way, we did not assume anything special about $\mathbb{M}$, and so whatever we could show about $\mathbb{M}$ must hold for all models.

Consider one more example. The argument $\forall x(Hx \,\&\, Jx) \,\therefore \forall x Hx$ is obviously valid. We can only show that the argument is valid by considering what must be true in every model in which the premise is true.

> Consider an arbitrary model $\mathbb{M}$ in which the premise $\forall x(Hx \,\&\, Jx)$ is true. The conjunction $Hx \,\&\, Jx$ is satisfied regardless of what is

assigned to x, so Hx must be also (by part 3 of the definition of satisfaction). As such, $\forall x Hx$ is satisfied by any variable assignment (by part 7 of the definition of satisfaction) and true in $\mathbb{M}$ (by the definition of truth). Since we did not assume anything about $\mathbb{M}$ besides $\forall x(Hx\,\&\,Jx)$ being true, $\forall x Hx$ must be true in any model in which $\forall x(Hx\,\&\,Jx)$ is true. So $\forall x(Hx\,\&\,Jx) \models \forall x Hx$.

Even for a simple argument like this one, the reasoning is somewhat complicated. For longer arguments, the reasoning can be insufferable. The problem arises because talking about an infinity of models requires reasoning things out in English. What are we to do? The answer won't surprise readers of the first half of the book: we'll make use of some formal proof systems. We have seen two kinds of proof systems for SL: the tree method, and natural deduction proofs. In the coming chapters, we'll extend both kinds of systems to QL as well.

Practice Exercises

⋆ **Part A** Determine whether each sentence is true or false in the model given.

$$\mathrm{UD} = \{\mathrm{Corwin,\ Benedict}\}$$
$$\mathrm{extension}(A) = \{\mathrm{Corwin,\ Benedict}\}$$
$$\mathrm{extension}(B) = \{\mathrm{Benedict}\}$$
$$\mathrm{extension}(N) = \varnothing$$
$$\mathrm{referent}(c) = \mathrm{Corwin}$$

1. Bc
2. $Ac \equiv \neg Nc$
3. $Nc \supset (Ac \lor Bc)$
4. $\forall x Ax$
5. $\forall x \neg Bx$
6. $\exists x(Ax\,\&\,Bx)$
7. $\exists x(Ax \supset Nx)$
8. $\forall x(Nx \lor \neg Nx)$
9. $\exists x Bx \supset \forall x Ax$

⋆ **Part B** Determine whether each sentence is true or false in the model given.

$$\mathrm{UD} = \{\mathrm{Waylan,\ Willy,\ Johnny}\}$$
$$\mathrm{extension}(H) = \{\mathrm{Waylan,\ Willy,\ Johnny}\}$$
$$\mathrm{extension}(W) = \{\mathrm{Waylan,\ Willy}\}$$
$$\mathrm{extension}(R) = \{<\mathrm{Waylan,\ Willy}>,<\mathrm{Willy,\ Johnny}>,<\mathrm{Johnny,\ Waylan}>\}$$
$$\mathrm{referent}(m) = \mathrm{Johnny}$$

1. $\exists x(Rxm \,\&\, Rmx)$
2. $\forall x(Rxm \lor Rmx)$
3. $\forall x(Hx \equiv Wx)$
4. $\forall x(Rxm \supset Wx)$
5. $\forall x\big[Wx \supset (Hx \,\&\, Wx)\big]$
6. $\exists x Rxx$
7. $\exists x \exists y Rxy$
8. $\forall x \forall y Rxy$
9. $\forall x \forall y(Rxy \lor Ryx)$
10. $\forall x \forall y \forall z\big[(Rxy \,\&\, Ryz) \supset Rxz\big]$

⋆ **Part C** Determine whether each sentence is true or false in the model given.

$$\text{UD} = \{\text{Lemmy, Courtney, Eddy}\}$$
$$\text{extension}(G) = \{\text{Lemmy, Courtney, Eddy}\}$$
$$\text{extension}(H) = \{\text{Courtney}\}$$
$$\text{extension}(M) = \{\text{Lemmy, Eddy}\}$$
$$\text{referent}(c) = \text{Courtney}$$
$$\text{referent}(e) = \text{Eddy}$$

1. Hc
2. He
3. $Mc \lor Me$
4. $Gc \lor \neg Gc$
5. $Mc \supset Gc$
6. $\exists x Hx$
7. $\forall x Hx$
8. $\exists x \neg Mx$
9. $\exists x(Hx \,\&\, Gx)$
10. $\exists x(Mx \,\&\, Gx)$
11. $\forall x(Hx \lor Mx)$
12. $\exists x Hx \,\&\, \exists x Mx$
13. $\forall x(Hx \equiv \neg Mx)$
14. $\exists x Gx \,\&\, \exists x \neg Gx$
15. $\forall x \exists y(Gx \,\&\, Hy)$

⋆ **Part D** Write out the model that corresponds to the interpretation given.

UD: natural numbers from 10 to 13
Ox: x is odd.
Sx: x is less than 7.
Tx: x is a two-digit number.
Ux: x is thought to be unlucky.
Nxy: x is the next number after y.

Part E Show that each of the following is contingent.

⋆ 1. $Da \,\&\, Db$
⋆ 2. $\exists x Txh$
⋆ 3. $Pm \,\&\, \neg \forall x Px$
4. $\forall z Jz \equiv \exists y Jy$
5. $\forall x (Wxmn \lor \exists y Lxy)$
6. $\exists x (Gx \supset \forall y My)$

⋆ **Part F** Show that the following pairs of sentences are not logically equivalent.

1. Ja, Ka
2. $\exists x Jx$, Jm
3. $\forall x Rxx$, $\exists x Rxx$
4. $\exists x Px \supset Qc$, $\exists x (Px \supset Qc)$
5. $\forall x (Px \supset \neg Qx)$, $\exists x (Px \,\&\, \neg Qx)$
6. $\exists x (Px \,\&\, Qx)$, $\exists x (Px \supset Qx)$
7. $\forall x (Px \supset Qx)$, $\forall x (Px \,\&\, Qx)$
8. $\forall x \exists y Rxy$, $\exists x \forall y Rxy$
9. $\forall x \exists y Rxy$, $\forall x \exists y Ryx$

Part G Show that the following sets of sentences are consistent.

1. {Ma, ¬Na, Pa, ¬Qa}
2. $\{Lee,\ Lef,\ \neg Lfe,\ \neg Lff\}$
3. $\{\neg(Ma \,\&\, \exists x Ax),\ Ma \lor Fa,\ \forall x(Fx \supset Ax)\}$
4. $\{Ma \lor Mb,\ Ma \supset \forall x \neg Mx\}$
5. $\{\forall y Gy,\ \forall x(Gx \supset Hx),\ \exists y \neg Iy\}$
6. $\{\exists x(Bx \lor Ax),\ \forall x \neg Cx,\ \forall x\big[(Ax \,\&\, Bx) \supset Cx\big]\}$
7. $\{\exists x Xx,\ \exists x Yx,\ \forall x(Xx \equiv \neg Yx)\}$
8. $\{\forall x(Px \lor Qx),\ \exists x \neg (Qx \,\&\, Px)\}$
9. $\{\exists z(Nz \,\&\, Ozz),\ \forall x \forall y(Oxy \supset Oyx)\}$
10. $\{\neg \exists x \forall y Rxy,\ \forall x \exists y Rxy\}$

Part H Construct models to show that the following arguments are invalid.

1. $\forall x(Ax \supset Bx)$, $\therefore$ $\exists x Bx$
2. $\forall x(Rx \supset Dx)$, $\forall x(Rx \supset Fx)$, $\therefore$ $\exists x(Dx \,\&\, Fx)$
3. $\exists x(Px \supset Qx)$, $\therefore \exists x Px$
4. $Na \,\&\, Nb \,\&\, Nc$, $\therefore$ $\forall x Nx$
5. Rde, $\exists x Rxd$, $\therefore$ Red
6. $\exists x(Ex \,\&\, Fx)$, $\exists x Fx \supset \exists x Gx$, $\therefore$ $\exists x(Ex \,\&\, Gx)$
7. $\forall x Oxc$, $\forall x Ocx$, $\therefore$ $\forall x Oxx$

8. $\exists x(Jx \mathbin{\&} Kx)$, $\exists x \neg Kx$, $\exists x \neg Jx$, $\therefore$ $\exists x(\neg Jx \mathbin{\&} \neg Kx)$

9. *Lab* $\supset$ $\forall x Lxb$, $\exists x Lxb$, $\therefore$ *Lbb*

Part I

1. Many logic books define consistency and inconsistency in this way: " A set $\{\Phi_1, \Phi_2, \Phi_3, \cdots\}$ is inconsistent if and only if $\{\Phi_1, \Phi_2, \Phi_3, \cdots\} \models (\Psi \mathbin{\&} \neg\Psi)$ for some sentence Ψ. A set is consistent if it is not inconsistent."

   Does this definition lead to any different sets being consistent than the definition on p. 70? Explain your answer.

⋆ 2. Our definition of truth says that a sentence Φ is TRUE IN $\mathbb{M}$ if and only if some variable assignment satisfies Φ in M. Would it make any difference if we said instead that Φ is TRUE IN $\mathbb{M}$ if and only if *every* variable assignment satisfies Φ in M? Explain your answer.

Chapter 10

QL Trees

In Chapter 9 we saw that the more structured models of QL make reasoning about all possible models rather complex. In SL, where a model was just an assignment of truth values to atomic sentences, reasoning about all models was relatively straightforward: we could simply survey them via truth tables. In QL there isn't really as simple a method for considering claims about validity or satisfiability as the truth table method. So we must rely more heavily on proof systems.

This chapter considers a tree method for QL. Like the tree method for SL discussed in Chapter 5, the QL tree method is a method that attempts to generate an interpretation satisfying a given set of sentences. When it succeeds, it shows a way to satisfy them; when it fails, it shows that they are unsatisfiable. As we did in the initial presentation of trees in Chapter 5, we'll begin by working through a couple of examples at an intuitive level, before laying out the formal tree rules.

10.1 Trees with fixed domains

Let's begin with a simplifying assumption. Let's suppose that we're only interested in models with particular fixed domain sizes. (By the time we get to the formal tree rules later in this chapter, we'll dispense with this assumption, but it's helpful for introducing the general idea.) Let's confine our attention to models that have two members in the UD. Consider whether there are any such models that satisfy the following QL sentences: $\{\forall x(Fx \supset Gx), \neg\exists xGx, \exists x(Fx \lor Gx)\}$. To evaluate this question, we put those sentences in the root of a tree.

1. $\forall x(Fx \supset Gx)$
2. $\neg\exists x Gx$
3. $\exists x(Fx \vee Gx)$

Let's start by considering the universal claim on line 1. A universal says that each of its instances are true. Since at the moment we are assuming that there are two objects in our UD, let's also assume we have just two names, a and b, corresponding to our two objects. So we can just write down the two instances as a linear development. We also put a check at line 1 to indicate that we've resolved that sentence.

1. $\forall x(Fx \supset Gx)$ ✓
2. $\neg\exists x Gx$
3. $\exists x(Fx \vee Gx)$
                   |
4. $Fa \supset Ga$
5. $Fb \supset Gb$

Next let's consider the negated existential at line 2. It says that each instance is false. So it develops into the negation of both instances, and we add a check mark at line 2.

1. $\forall x(Fx \supset Gx)$ ✓
2. $\neg\exists x Gx$ ✓
3. $\exists x(Fx \vee Gx)$
                   |
4. $Fa \supset Ga$
5. $Fb \supset Gb$
                   |
6. $\neg Ga$
7. $\neg Gb$

There is one more quantified sentence to resolve, on line 3, but before we do that, let's handle the conditionals on lines 4 and 5. We do these exactly the same way we handled conditionals in SL trees. They each branch into the negation of the antecedent and the consequent; the latter branch closes each time.

1. $\forall x(Fx \supset Gx)$ ✓
2. $\neg \exists x Gx$ ✓
3. $\exists x(Fx \vee Gx)$

$|$

4. $Fa \supset Ga$ ✓
5. $Fb \supset Gb$ ✓

$|$

6. $\neg Ga$
7. $\neg Gb$

8. $\neg Fa$ Ga
                    $\times$
                  $6, 8$
9. $\neg Fb$ Gb
          $\times$
        $7, 9$
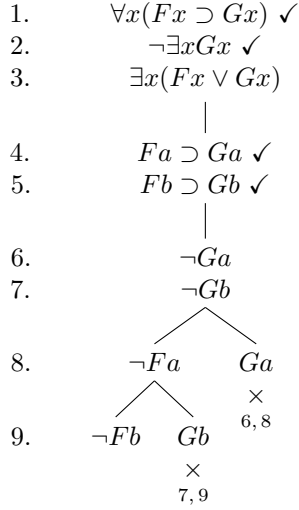
Now let's look at line 3. It is an existential claim, so it says that at least one of its instances are true. Again, we're assuming for now that there are only two instances. So we branch into them those two disjunctions at line 10. After that, we apply the approach to disjunctions from SL trees, and the tree closes.

1. $\forall x(Fx \supset Gx)$ ✓
2. $\neg \exists x Gx$ ✓
3. $\exists x(Fx \vee Gx)$

$|$

4. $Fa \supset Ga$ ✓
5. $Fb \supset Gb$ ✓

$|$

6. $\neg Ga$
7. $\neg Gb$

8. $\neg Fa$ Ga
                         $\times$
                       $6, 8$
9. $\neg Fb$ Gb
                 $\times$
              $7, 9$
10. $Fa \vee Ga$ $Fb \vee Gb$

11. Fa Ga Fb Gb
     $\times$ $\times$ $\times$ $\times$
   $8, 11$ $6, 11$ $9, 11$ $7, 11$
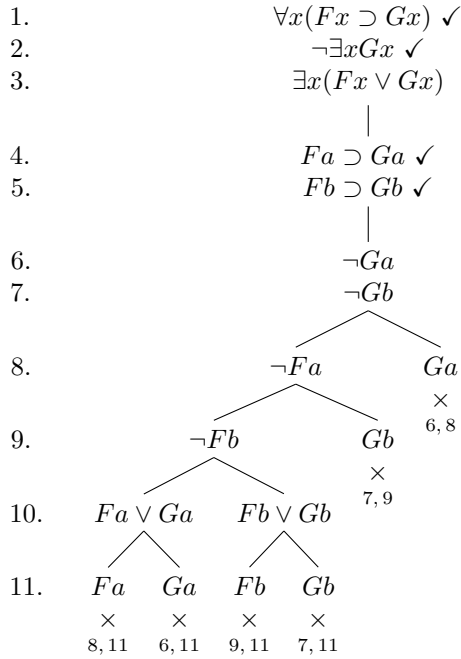
The closed tree indicates that our attempt to find a model satisfying the root has failed. We're not quite in a position to say that we've proven the root

entirely unsatisfiable, because we've been assuming for simplicity that we're only interested in models that have two members of the UD. But we've shown at least that it's impossible to satisfy the root with any model with a two-object domain. We'll consider a more generalized procedure soon.

Before moving on to the official tree rules, let's look at one more example that assumes a two-object domain.

Is there any model with a two-object UD that satisfies $\{\forall x Fxa, \forall y(Gy \supset \neg Fya)\}$? To find out, we put both sentences in the root of a tree, and continue as before. We start by taking both instances of each universal claim, then process the two conditionals. This tree remains open.

1. $\qquad\qquad\quad \forall x Fxa$ ✓
2. $\qquad\quad \forall y(Gy \supset \neg Fya)$ ✓
   $\qquad\qquad\qquad\quad |$
3. $\qquad\qquad\quad\; Faa$
4. $\qquad\qquad\quad\; Fba$
   $\qquad\qquad\qquad\quad |$
5. $\qquad\qquad\; Ga \supset \neg Faa$ ✓
6. $\qquad\qquad\; Gb \supset \neg Fba$ ✓

7. $\qquad\quad \neg Ga \qquad \neg Faa$
   $\qquad\qquad\qquad\qquad\quad \times$
   $\qquad\qquad\qquad\qquad\quad 3, 7$
8. $\qquad \neg Gb \quad \neg Fba$
   $\qquad\quad \uparrow \qquad \times$
   $\qquad\qquad\qquad 4, 8$
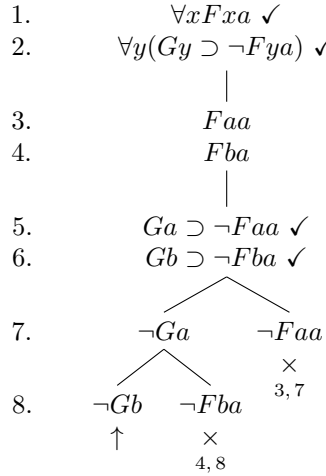
As in SL trees, a completed QL tree with an open branch describes a way of satisfying the root. We're assuming a two-object UD; for simplicity, let's let our objects be the letters 'a' and 'b', and have 'a' be the referent of the QL name a, and 'b' be the referent of the QL name b. We look to the two QL atoms and two negated QL atoms in the branch; they indicate what needs to be included in the extension of each predicate in our model. Since Faa is in the branch, we need to include <a, a> in the extension of F. Since $\neg Ga$ is in the branch, we need for a *not* to be in the extension of G.

Here is a partial model that suffices to satisfy the root of the above tree:

$$\mathrm{UD} = \{a, b\}$$

$$\text{extension}(G) = \begin{array}{c|c} & G \\ \hline \mathbf{a} & 0 \\ \mathbf{b} & 0 \end{array}$$

$$\text{extension}(F) = \begin{array}{c|cc} Fxy & \mathbf{a} & \mathbf{b} \\ \hline \mathbf{a} & 1 & - \\ \mathbf{b} & 1 & - \end{array}$$

Notice that we've put blanks for two of the cells in the extension of Fxy. That is to indicate that it doesn't matter whether or not <a, b> or <b, b> are included in the extension of Fxy. (This is also why neither Fab, Fbb, $\neg Fab$, nor $\neg Fbb$ appeared in the open branch.) So the tree method for QL is similar to that for SL in this respect as well. An open branch in a completed tree describes a way to satisfy the root.

10.2 Generalizing the tree method

Trees are supposed to show something about *all* possible models, not only those with two-object UDs. So we don't want our official tree rules to build in the idea that there are just two objects. For a three-object UD, we'd want our universal statements to develop into all three instances, and we'd want our existentials to branch into the three instances. For a ten-object UD like the one described on page 165, we'd need to give ten instances. For an infinite UD like the set of all natural numbers, we'd need an *infinite* number of developments. Obviously such rules would be impractical, to say the least. Worse than that, we often won't know at the start of the process what size domain we should be thinking of. So we need a version of the tree rules that is flexible as to domain size.

In order to state our rule formally, we need one new bit of formalism. For any QL wff Φ, a constant c, and variable χ, define $\Phi \boxed{\chi \Rightarrow c}$ to mean the wff that we get by replacing every occurrence of χ in Φ with c. So for example,

$$Fx \boxed{x \Rightarrow a} = Fa$$

For any quantified QL sentence $\forall \chi \Phi$ or $\exists \chi \Phi$, we call $\Phi \boxed{\chi \Rightarrow c}$ a SUBSTITUTION INSTANCE of $\forall x \Phi$ and $\exists x \Phi$, and c is called the INSTANTIATING CONSTANT.

This, more formally, is the same notion we were working with in our discussion of trees for models with two-object UDs. For example:

▷ $Aa \supset Ba$, $Af \supset Bf$, and $Ak \supset Bk$ are all substitution instances of $\forall x(Ax \supset Bx)$; the instantiating constants are a, f, and k, respectively.

▷ Raj, Rdj, and Rjj are substitution instances of $\exists z Rzj$; the instantiating constants are a, d, and j, respectively.

With this terminology in hand, we can begin with our official tree rules.

10.3 Existentials

This will be our official tree rule for existentials:

$$\exists \chi \Phi \;\checkmark a$$
$$|$$
$$\Phi \boxed{\chi \Rightarrow a} \qquad \text{where } a \text{ is new}$$

Just as Φ stands in for any sentence of QL, the χ is meant to stand in for any variable, and a stands in for any name. The requirement that the name be *new* means that the name chosen for the substitution instance must be a name that has not appeared anywhere in the tree so far.

What is the rationale for this restriction? The existential claim says that some substitution instance is true, but it doesn't say which one it is. If we knew how many objects we were looking at in the UD, we could make a new branch for each object — that's what we did in the examples above — but here we see a different way to model this kind of flexibility. Remember, there is no prohibition in QL of two different names referring to the very same object. So by using an instance involving a new name, we are remaining open-minded about whether it is also a name for some object we've already been discussing.

For example, consider this tree:

1. Fa
2. $\exists x \neg Fx$ $\checkmark b$
               $|$
3. $\neg Fb$ 2 $\exists$

The appropriate substitution instance of the existential on line 2 is Fb, not Fa, because a is a name that is already in use. We want to say that *something* is F, not necessarily that a is. Had we taken the a instance, the tree would have closed, even though the root is satisfiable. Note also that in marking the existential as having been processed, we also make a note of the instantiating constant. That's why there is a b next to the check mark in line 2.

10.4 Universals

Here is the tree rule for universals.

$$\forall \chi \Phi \; \backslash a$$

$$\Phi \;\boxed{\chi \Rightarrow a} \qquad \text{for any } a$$
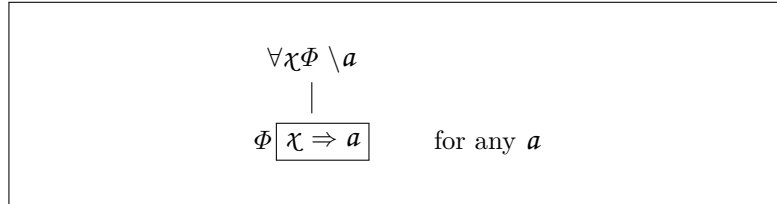
The universal rule, like the existential rule, invites you to take an instance. But unlike the existential rule, we don't limit ourselves to instances with new names. The other major difference is that one can use this rule multiple times. That makes this rule different from all the other tree rules we've seen so far. You're not necessarily finished processing a universal sentence after you've performed the rule once. We mark this by indicating a \ instead of a ✓ next to the universal sentence.

Here is an example illustrating the importance of taking multiple instances. Here is a partially completed tree.

1. $\forall x (Fx \,\&\, Gx) \; \backslash a$
2. $\neg Fa \lor \neg Gb$ ✓

3. $Fa \,\&\, Ga$ ✓   1 $\forall$

4. Fa 3 &
5. Ga

6. $\neg Fa$ $\neg Gb$ 2 $\lor$
      ×
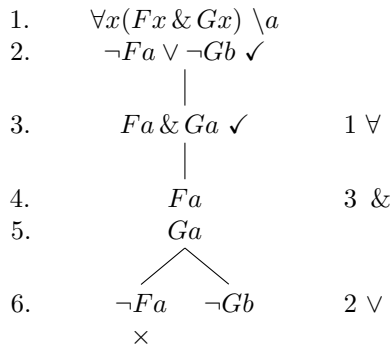
The root of this tree is pretty obviously unsatisfiable, but so far, the tree hasn't closed. On line 3 we take the a instance of the universal at line 1, but the tree won't close until we take the b instance as well. We can do so at any time. In this way of developing the tree, we take that instance at line 7, and then the tree will close:

1. $\forall x(Fx \,\&\, Gx) \;\backslash a, b$
2. $\neg Fa \lor \neg Gb$ ✓
                    |
3. $Fa \,\&\, Ga$ ✓   1 $\forall$
                    |
4. Fa 3 &
5. Ga

6. $\neg Fa$ $\neg Gb$ 2 $\lor$
               × |
7. $Fb \,\&\, Gb$ ✓   1 $\forall$
                    |
8. Fb 7 &
9. Gb
                    ×

Of course, we could also have taken the b instance at the same time as we took the a instance on line 3, and the tree would have worked just as well. But taking things in this order illustrates that sometimes we can go back to universal claims that have already been considered, to take an additional instance. (Later on, we will see some examples of trees that *require* this kind of backtracking.)

10.5 Negated existentials

Negated existential claims are quite similar to universals. They say that no instance is true, which is another way of saying that every instance is false. So like the universal rule, the negated existential rule will allow us to take as many instances as we like; this time, we take the instances of the *negation* of the wff in the scope of the quantifier.

$\neg\exists\chi\,\Phi \;\backslash a$
|
$\neg\Phi\;\boxed{\chi \Rightarrow a}$ for any a

For example, if $\neg\exists xFx$ is in the tree, one can use the negated existential rule to develop a tree branch with $\neg Fa$, $\neg Fb$, etc.

10.6 Negated universals

Conversely, a negated universal is similar to an existential. It says that not all instances are true, so at least one instance is false. So like the existential rule, we only perform it once (marking it with a check), and ensuring that we use a new name, so as not to assume anything in particular about the instance chosen.

$$\neg\forall\chi\Phi \checkmark a$$
$$|$$
$$\neg\Phi \boxed{\chi \Rightarrow a} \qquad \text{where } a \text{ is } new$$

10.7 Whither branching?

When we learned the tree rules for sentential connectives, there was an important distinction between *linear* rules and *branching* rules. We retained this when we assumed two-object domains in §10.1, where universal quantifiers had linear rules, and existentials had branching rules. Our new quantifier rules, however, all have the same linear shape. Nothing branches. Why not?

In SL, and for QL models where finite domains were specified in advance, we used branching to represent the particular possible ways one could satisfy the formula above. For some interpretation I to satisfy $P \vee Q$, either $I(P) = 1$ or $I(Q) = 1$. If we assume there are only two objects, a and b, then in order for I to satisfy $\exists x F x$, I must either have a in the extension of F or have b in the extension of F. So we can branch the tree to represent those two possibilities.

But if we do not assume anything in particular about the UD, we can't just list all the possible ways to satisfy an existential (or a negated universal). We'd need as many branches as there are objects in the UD. So instead we just take one instance, but give it a new name, which may or may not be co-referential with another name we've already considered. The novelty of the name is playing the functional role of branching.

10.8 The other development rules

So far this chapter we've introduced four tree development rules for QL: we have rules for universals, existentials, negated universals, and negated existentials.

There are nine more resolution rules, but they are all familiar: they are the same nine resolution rules given in Chapter 5. We'll use all the same resolution rules for conjunction, negated conjunction, disjunction, negated disjunction, conditional, negated conditional, biconditional, negated biconditional, and double negation as we did before. They were introduced on pp. 77–81.

10.9 Branch closure rules

Our branch closure rules also remain unchanged. A branch closes if and only if it contains some sentence Φ along with its negation, $\neg\Phi$. We mark closed branches with an '×'.

10.10 Tree completion rules

The completion rules, however, require a modification. The SL tree completion rule, given in §5.6, said that a branch was complete if every sentence, other than atoms and negated atoms, has been resolved. In the QL tree system we need for our tree completion rules to take account of the fact that some sentences need to be acted on multiple times. The universal rule, for example, lets one take whatever instance one likes; we need our completion rules to ensure that we've taken *enough* instances.

For example, this should not count as a completed tree:

1. $\forall x(Fx \supset Ga) \setminus a$
2. $\neg Ga$
3. Fb
    |
4. $Fa \supset Ga$ ✓
5. $\neg Fa \quad Ga$
    ×
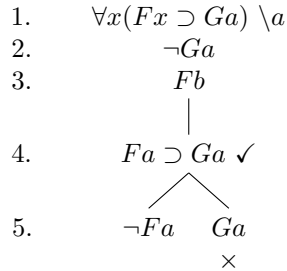
Our universal rule can be processed multiple times. This tree has only taken the a instance of $(Fx \supset Ga)$; if it also took the b instance, the tree would close:

1. $\forall x(Fx \supset Ga) \ \backslash a, b$
2. $\neg Ga$
3. Fb
                      |
4. $Fa \supset Ga$ ✓

5. $\neg Fa$ Ga
                              ×
               |
6. $Fb \supset Ga$ ✓

7. $\neg Fb$ Ga
           ×   ×
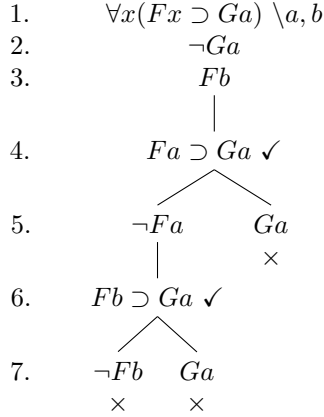
We don't want this tree to count as complete until it's taken both the *a* and the *b* instance. The universal says that every instance must be true; we don't ensure that this is satisfied if only some instances have been taken.

If we knew how many names our model used, we could simply require that every instance be taken. But we don't know in advance how many names our models are going to use. QL trees construct their interpretations as they go. (Remember that resolving existentials and negated universals involves introducing new names.) Here is the requirement we want: we require that we've taken the instance corresponding to *every name in the branch*. If, after taking an instance or two, we go on to introduce a new name within the same branch, that branch isn't complete until we've taken that instance as well. Once again, this is why we use the \ instead of a ✓. We can't be sure that we're finished with that line until we've done the rest of the tree.

A wff is RESOLVABLE if it is the kind of wff that gets a check mark upon processing it. (That's everything except for atoms, negated atoms, universals, and negated existentials.) A wff is GENERAL if it is a universal or a negated existential. Here is our new tree completion rule:

> A branch is COMPLETE if and only if either (i) it is closed, or (ii) every resolvable sentence in the branch has been resolved, and for every general sentence and every name *a* in the branch, the *a* instance of that general sentence has been taken.
> A tree is complete if and only if every branch in that tree is complete.

10.11 Resolution order

As in the SL tree system, the rules do not indicate any particular resolution order. But — also as in the SL system — some strategies are going to be more efficient than others. In fact, given the increased complexity of our rules and the fact that some resolutions introduce new names, it is even more important, from an efficiency perspective, than it used to be to employ sensible strategies. It is still a good idea to do linear rules before branching rules, and when possible, to perform a branching rule when you can see that one branch will close right away. Now, we can use similar strategies to take instances from general sentences in a strategic way. If you're not sure which instance to take first, look a step ahead and see if one of the instances will close a branch.

Here is a simple tree illustrating the point. Let's consider whether $Gb \supset \forall x \neg Fx$ and $\exists x Fx$ entail $\neg Gb$. We begin by putting the first two sentences and the negation of the third in the root of a tree:

1. $Gb \supset \forall x \neg Fx$
2. $\exists x Fx$
3. $\neg\neg Gb$

Ordinarily, it is efficient not to branch too early. But in this case, if we look ahead a step, we can see that resolving the conditional on line 1 will close one branch immediately. So that's a good place to start.

1. $Gb \supset \forall x \neg Fx$ ✓
2. $\exists x Fx$
3. $\neg\neg Gb$

4. $\neg Gb$ $\forall x \neg Fx$ $1 \supset$
        ×
       $3, 4$

From here we have three choices to consider: lines 2, 3, and 4. Line 3 pretty obviously won't get us anywhere interesting, so let's consider either the existential at line 2 or the universal at line 4. It's usually a good rule of thumb to do the new instances first — that way we'll have more information when we make decisions about which instances to take for the general rules. So let's take the a instance of line 2.

1. $Gb \supset \forall x \neg Fx$ ✓
2. $\exists x Fx$ ✓a
3. $\neg\neg Gb$

4. $\neg Gb$ $\forall x \neg Fx$ $1 \supset$
              ×
            3, 4
5. Fa $2 \exists$

Now we turn our attention to the universal at line 4. We can take whatever instance we want. Let's be intelligent about it. The *b* instance wouldn't do anything interesting; Fb won't interact with that $\neg\neg Gb$ in any helpful way. But the *a* instance will close the tree. So we take that one.

1. $Gb \supset \forall x \neg Fx$ ✓
2. $\exists x Fx$ ✓a
3. $\neg\neg Gb$

4. $\neg Gb$ $\forall x \neg Fx$ \\a $1 \supset$
            ×
           3, 4
5. Fa $2 \exists$

6. $\neg Fa$ $4 \forall$
                   ×
                  5, 6

Planning ahead can help make your trees more efficient. The general principle you should keep in mind is, if you have a choice for what resolvable sentence to resolve, or what instance of a general sentence to take, make a choice that will allow you to close branches more quickly.

10.12 Infinite trees

In the SL tree system, every development of the tree took us closer to a completed tree, because we checked sentences above as resolved, and wrote simpler sentences in one or two branches below. SL trees are guaranteed to be completed in a finite number of steps — they will either close, or they will reach a point where everything resolvable has been resolved.

Because of the way our QL system works, it is possible for trees to continue indefinitely without closing. General sentences require that instances be taken for all names in the branch; but the rules for existentials and negated universals require that new names be introduced. Often, we can introduce the new names before dealing with the general instances, as in this completed tree:

1. $\exists x F x$ ✓ a
2. $\forall x G x$ ∖ a
              |
3. Fa 1 ∃
              |
4. Ga 2 ∀
              ↑

In this tree, we introduce the name a for the existential, then take the instance for the one and only name in the branch for the universal, and the tree is complete. But things won't always be so simple. Consider a tree with this single sentence in its root: $\forall x \exists y R x y$. In this case, we must begin by performing the universal rule, as our one and only sentence is a universal. This rule allows us to take an instance with any name. Ordinarily, we use the names that already exist in the branch, but since there are no names, we'll just take the a instance from the start, putting that instance — an existential — on line 2.

To resolve that existential, we must use a new name. So on line 3, we take the b instance from line 2.

1. $\forall x \exists y R x y$ ∖ a
              |
2. $\exists y R a y$ ✓ b 1 ∀
              |
3. Rab 2 ∃

This tree has not closed, but note that it is not complete. We haven't taken the b instance of the universal on line 1. When we do so, we will find ourselves with another existential on line 4; it in turn requires an instance involving a new name, and the process will continue indefinitely.

1. $\forall x \exists y Rxy \;\; \backslash a, b, c$

             |

2. $\exists y Ray \;\checkmark b$ 1 $\forall$

             |

3. Rab 2 $\exists$

             |

4. $\exists y Rby \;\checkmark c$ 1 $\forall$

             |

5. Rbc 4 $\exists$

             |

6. $\exists y Rcy \;\checkmark d$ 1 $\forall$

             |

7. Rcd 5 $\exists$

             |

8. $\vdots$

As the pattern makes clear, this tree will never close, but it will also never be complete. It will just keep taking instances of an existential with new names, then taking those instances of the universal, which in turn require a new existential, and so on. (Eventually we will run out of letters, but we have subscripts available too, so we'll never run out of names.) Because the tree will never close, its open branch is describing a model that satisfies the root, but the model suggested is an infinite one. Line 3 tells us that <a, b> should be in the extension of R; line 5 says that <b, c> should too. There is a clear pattern that tells us what R's extension must be like:

  extension(R) = {<a, b>, <b, c>, <c, d>, <d, e>, ...}

Or we can represent it graphically, thus:

$$\text{extension}(R) = \begin{array}{c|ccccc} Rxy & a & b & c & d & \dots \\ \hline a & - & 1 & - & - & - \\ b & - & - & 1 & - & - \\ c & - & - & - & 1 & - \\ d & - & - & - & - & 1 \\ \vdots & - & - & - & - & - \end{array}$$

The blanks indicate that it doesn't matter whether those pairs fall under the extension of R or not.

What this suggests is that if one had a UD comprising an infinite list of objects, with each object related via R to the next item in the list, this model would satisfy the root of the tree. For example, suppose that our UD were the set of all

natural numbers, and that we interpreted Rxy as $x + 1 = y$. For every number x, there is a number y that is equal to $x + 1$.

Note, however, that this is only one of many possible ways to construct a model meeting the constraint described. In fact, we don't necessarily require that our UD be infinite. Our tree ended up using an infinite number of names, but there's no rule requiring that each name be assigned to a unique object.

The extension of R requires that certain pairs be included. Note that it doesn't require that any pairs be excluded. There are no 0s in the chart above. The blanks mean it doesn't matter whether you include them. A model that included every ordered pair in the extension of R would be an example of the kind of model indicated:

Rxy	a	b	c	d	...
a	1	1	1	1	1
b	1	1	1	1	1
c	1	1	1	1	1
d	1	1	1	1	1
⋮	1	1	1	1	1

Once we put it this way, we can see that the R predicate needn't draw any distinctions between the various objects in the domain. Each object could be the same, as far as R is concerned. So we could even offer a simpler model, with a UD comprising a simple object that is R-related to itself.

$$\text{UD} = \{a\}$$
$$\text{extension}(R) = \{<a, a>\}$$

When a tree continues indefinitely, the root is satisfiable; you can describe a model for it using an infinite domain, taking advantage of the pattern indicated by the infinite branch; or you can often find a way to simplify it into a model with a finite UD.

Let's consider another example of an infinite tree. This time, we'll use names with subscripts. Everything is straightforward up until line 6; at line 7, we need to go back and take a second instance of line 2, and the infinite cycle begins.

1. $\quad\quad \exists x \forall y \exists z(\neg Rxy \,\&\, Rzx) \;\checkmark a_1$

   $\quad\quad\quad\quad\quad |$

2. $\quad \forall y \exists z(\neg Ra_1 y \,\&\, Rza_1) \;\backslash a_1, a_2, a_3 \quad\quad 1\ \exists$

   $\quad\quad\quad\quad\quad |$

3. $\quad\quad \exists z(\neg Ra_1 a_1 \,\&\, Rza_1) \;\checkmark a_2 \quad\quad\quad 2\ \forall$

   $\quad\quad\quad\quad\quad |$

4. $\quad\quad\quad \neg Ra_1 a_1 \,\&\, Ra_2 a_1 \;\checkmark \quad\quad\quad 3\ \exists$

   $\quad\quad\quad\quad\quad |$

5. $\quad\quad\quad\quad\quad \neg Ra_1 a_1 \quad\quad\quad\quad\quad 4\ \&$
6. $\quad\quad\quad\quad\quad\; Ra_2 a_1$

   $\quad\quad\quad\quad\quad |$

7. $\quad\quad \exists z(\neg Ra_1 a_2 \,\&\, Rza_1) \;\checkmark a_3 \quad\quad\quad 2\ \forall$

   $\quad\quad\quad\quad\quad |$

8. $\quad\quad\quad \neg Ra_1 a_2 \,\&\, Ra_3 a_1 \;\checkmark \quad\quad\quad 8\ \exists$

   $\quad\quad\quad\quad\quad |$

9. $\quad\quad\quad\quad\quad \neg Ra_1 a_2 \quad\quad\quad\quad\quad 9\ \&$
10. $\quad\quad\quad\quad\quad\; Ra_3 a_1$

    $\quad\quad\quad\quad\quad |$

11. $\quad\quad \exists z(\neg Ra_1 a_3 \,\&\, Rza_1) \;\checkmark a_4 \quad\quad\quad 2\ \forall$

    $\quad\quad\quad\quad\quad |$

12. $\quad\quad\quad \neg Ra_1 a_3 \,\&\, Ra_4 a_1 \;\checkmark \quad\quad\quad 11\ \exists$

    $\quad\quad\quad\quad\quad |$

13. $\quad\quad\quad\quad\quad \neg Ra_1 a_3 \quad\quad\quad\quad\quad 9\ \&$
14. $\quad\quad\quad\quad\quad\; Ra_4 a_1$

    $\quad\quad\quad\quad\quad |$

15. $\quad\quad\quad\quad\quad \vdots$

If we continued drawing the tree, we'd take the a_4 instance of the universal at line 2 on line 15, then introduce a new name by resolving that universal. The tree will continue forever, again with a clear pattern. Lines 5, 9, 13, etc. tell us that for every integer i, $Ra_1 a_i{=}0$. Lines 6, 10, 14, etc. tell us that for every $i > 1$, $Ra_i a_1{=}1$. Or in chart form:

Rxy	a_1	a_2	a_3	a_4	$\ldots$
a_1	0	0	0	0	0
a_2	1	-	-	-	-
a_3	1	-	-	-	-
a_4	1	-	-	-	-
$\vdots$	1	-	-	-	-

This chart describes the extension of R that, as part of a model with the infinite

UD $\{a_1, a_2, a_3, \ldots\}$, would satisfy $\exists x \forall y \exists z (\neg Rxy \,\&\, Rzx)$. As in the previous example, we do not *need* an infinite model to satisfy this sentence; the extension of R here can be collapsed into a finite one. In the previous example, our extension required no distinctions between any of the objects; here, we do have some required differences. $Ra_1a_1 = 0$, but $Ra_2a_1 = 1$. So our model is treating a_1 and a_2 differently. But it requires no further distinctions than that. Every object after a_2 can be the same as a_2, as far as this model is concerned. So a two-object model would also suffice to satisfy this sentence. Here is a graphical representation of the denotation for R:

Rxy	a_1	a_2
a_1	0	0
a_2	1	-

Or, describing the model in question with sets and n-tuples:

$$\text{UD} = \{a_1, a_2\}$$
$$\text{extension}(R) = \{<a_2, a_1>\}$$

The reasoning involved in collapsing models with infinite domains into finite ones centrally involves reasoning about what distinctions are required: do we need to suppose that the objects named are *different objects*, or can we treat it as a case of multiple names for the *same thing*. In Chapter 12, we'll foreground this kind of question in more detail, and introduce a way to talk about it in QL.

Note that, for purposes of answering questions about entailment, you needn't go through the reasoning that allows you to move from infinite models to finite ones that would serve just as well. The tree method, most fundamentally, is a question about whether there is any possible model of the root. If a tree continues infinitely, then it is answering that question: yes, there is an infinite model for it. It's not necessary, for the question of whether there is any model satisfying the root, to determine whether there is a *finite* model that does so.

10.13 Common student errors

10.13.1 Using old names

The most frequent errors for trees in QL involve ignoring the restrictions on names involved in taking instances. Particular sentences (existentials and negated universals) require a *new* name — if the name appears anywhere in the branch at the time the existential is taken (including below it), it is not suitable for substitution via these rules.

For example, this tree is a mistake:

1. $\forall x \exists y \neg Rxy$ \a
2. Rab
      |
3. $\exists y \neg Ray$ ✓b 1 $\forall$
      |
4. $\neg Rab$ 3 $\exists$
     ×
     2, 4

Line 3 here is fine; but at line 4, this tree takes the b instance of the existential on line 3, even though b appeared in line 2. The existential rule requires a new name. The correct version of this tree will remain open. (It will extend infinitely.)

10.13.2 Main Connectives

Always work on the main connectives. If you have a sentence like

$$\forall x \exists y \neg Ryx \,\&\, (Fa \supset Fb)$$

don't use the universal rule. This sentence is a conjunction, so you must perform the conjunction rule on it. That will give you a universal, at which point you can take an instance.

Practice Exercises

⋆ **Part A** Use a tree to test whether the following sentences are tautologies. If they are not tautologies, describe a model on which they are false.

1. $\forall x \forall y (Gxy \supset \exists z Gxz)$
2. $\forall x Fx \vee \forall x (Fx \supset Gx)$
3. $\forall x (Fx \supset (\neg Fx \supset \forall y Gy))$
4. $\exists x (Fx \vee \neg Fx)$
5. $\exists x Jx \equiv \neg \forall x \neg Jx$
6. $\forall x (Fx \vee Gx) \supset (\forall y Fy \vee \exists x Gx)$

⋆ **Part B** Use a tree to test whether the following argument forms are valid. If they are not, give a model as a counterexample.

1. Fa, Ga, $\therefore$ $\forall x (Fx \supset Gx)$
2. Fa, Ga, $\therefore$ $\exists x (Fx \,\&\, Gx)$

3. $\forall x \exists y Lxy$, $\therefore$ $\exists x \forall y Lxy$
4. $\exists x (Fx \,\&\, Gx)$, $Fb \equiv Fa$, $Fc \supset Fa$, $\therefore$ Fa
5. $\forall x \exists y Gyx$, $\therefore$ $\forall x \exists y (Gxy \lor Gyx)$

Part C Translate each argument into QL, specifying a UD, then use a tree to evaluate the resulting form for validity. If it is invalid, give a model as a counterexample.

1. Every logic student is studying. Deborah is not studying. Therefore, Deborah is not a logic student.
2. Kirk is a white male Captain. Therefore, some Captains are white.
3. The Red Sox are going to win the game. Every team who wins the game will be celebrated. Therefore, the Red Sox will be celebrated.
4. The Red Sox are going to win the game. Therefore, the Yankees are not going to win the game.
5. All cats make Frank sneeze, unless they are hairless. Some hairless cats are cuddly. Therefore, some cuddly things make Frank sneeze.

Chapter 11

Soundness and Completeness for QL Trees

In Chapter 6 we proved that our SL tree method works the way it is supposed to: any tree that closed was guaranteed to have a root that was unsatisfiable in SL, and any completed tree that remained open was guaranteed to describe an SL model that satisfies the root. These two theorems together are called *soundness* and *completeness*.

SOUNDNESS: If a tree closes, that guarantees that its root is unsatisfiable. In other words:

$$\mathcal{X} \vdash \bot \Rightarrow \mathcal{X} \models \bot$$

COMPLETENESS: If a root is unsatisfiable, that guarantees that the tree will close. In other words:

$$\mathcal{X} \models \bot \Rightarrow \mathcal{X} \vdash \bot$$

These definitions were first given on pages 89 and 97. They are equally applicable to our extended tree system for QL. The system introduced in Chapter 10 is also sound and complete. The rules are designed in a way so as to guarantee that if the tree closes, there is no model for the root (sound), and to guarantee that an open branch describes a model for the root (complete). The proofs for these two theorems are structurally very similar to the proofs given in Chapter 6.

11.1 Soundness

If our tree method is sound, then there is no possible set of QL sentences $\mathcal{X}$ that are satisfiable, where a tree with root $\mathcal{X}$ closes. Not every possible tree method is sound. For example, suppose we dropped the requirement, for a negated universal, that one take an instance with a new name.

In other words, suppose we replaced this rule (introduced on page 189)

$$\neg\forall\chi\Phi \;\checkmark\, a$$
$$|$$
$$\neg\Phi\;\boxed{\chi \Rightarrow a} \qquad \text{where } a \text{ is } new$$

with this alternate one:

$$\neg\forall\chi\Phi \;\checkmark\, a$$
$$|$$
$$\neg\Phi\;\boxed{\chi \Rightarrow a} \qquad \text{for } any \; a$$

If we had this rule, there would be counterexamples to soundness — trees with satisfiable roots, which nevertheless closed. Here is an example:

1. Fa
2. $\neg\forall x Fx$
         |
3. $\neg Fa$ 2 alt˙¬∀
       ×
      1, 3

This tree is a counterexample to the soundness of the alternate tree system just described. To prove that our system is sound is to prove that our actual rules do not allow for this kind of tree. To prove this, we'll begin by assuming that $\mathcal{X}$ is satisfiable, and demonstrate from that assumption that at least one branch of any tree that develops according to our rules will remain open. As in the case of the parallel discussion in Chapter 6, our proof will be a recursive one. We will demonstrate that, if the tree *starts* with a set of satisfiable sentences, then, for each legal way the tree may develop, at least one branch will continue to have a set of satisfiable sentences. This is effectively to show that such a branch will never close, since branches only close when they contain some sentences Φ and $\neg\Phi$, which are of course never jointly satisfiable.

Suppose, then, we have some satisfiable set of sentences $\mathcal{X}$ in the root of a tree. If the root is satisfiable, then there is some model that satisfies it. Call this interpretation $\mathcal{I}$. We prove that, if our tree follows the rules given in Chapter 10, then $\mathcal{I}$ doesn't just satisfy the root — it satisfies every sentence in any completed branch. We will prove this recursively.

11.1.1 Root

We assume that the tree begins with a satisfiable root. Given this assumption, $\mathcal{I}$ is just our name for one of the interpretations we are assuming must exist. So $\mathcal{I}$ trivially satisfies everything in the root.

Now we must prove, for each possible way of developing the tree, that if the sentences in the branch we *begin* with are satisfiable, then the sentences we have after applying the rule are satisfiable too. There are thirteen possible ways a tree can develop, corresponding to the thirteen kinds of non-atomic sentences in QL, each of which has a particular processing rule. The thirteen kinds of sentences are:

- ▷ double negation
- ▷ conjunction
- ▷ negated conjunction
- ▷ disjunction
- ▷ negated disjunction
- ▷ conditional
- ▷ negated conditional
- ▷ biconditional
- ▷ negated biconditional
- ▷ existential
- ▷ negated existential
- ▷ universal
- ▷ negated universal

Fortunately, we've already proven what we need to prove for the first nine items on the list. Our QL tree method uses all the same rules as the SL method did

for the sentential connectives; we proved in §6.4.2, for each of those rules, that it has the key property: if some interpretation $\mathcal{I}$ satisfies what comes above the rule, then the development below it is also satisfiable. (Indeed, we proved there that the very same interpretation, $\mathcal{I}$, satisfied it.)

So to extend our soundness proof to our QL system, we need only prove the same thing for the four rules for quantifiers. This is the project of the next four subsections.

11.1.2 Existentials

Suppose some satisfiable set of QL sentences $\mathcal{X}$ is developed according to the existential rule:

$$\exists \chi \Phi \; \checkmark a$$
$$|$$
$$\Phi \boxed{\chi \Rightarrow a} \qquad \text{where } a \text{ is } new$$

We assume that $\mathcal{I}$ models $\mathcal{X}$, which includes some existential $\exists \chi \Phi$. We want to prove that there is a model for the expanded branch which comprises both $\mathcal{X}$ and $\Phi \boxed{\chi \Rightarrow a}$, i.e., $\mathcal{X} \cup \Phi \boxed{\chi \Rightarrow a}$ (the set containing $\mathcal{X}$ and the new development sentence too). Unlike in the parallel proof for the sentential rules, we cannot be sure that $\mathcal{I}$ itself satisfies our new development, because our new development introduces a new name; we cannot assume that $\mathcal{I}$ included any assignment for the new name a. Nor, if it did, are we sure that the object a denotes satisfies Φ. But we *can* be assured that $\mathcal{I}$ can be expanded into a new, similar interpretation, $\mathcal{I}^*$, which does include a. Moreover, since we know that $\mathcal{I}$ satisfied the existential $\exists \chi \Phi$, we know that there was some object in $\mathcal{I}$'s domain that satisfied Φ. So it will be possible to construct our new interpretation $\mathcal{I}^*$ so that it includes the new name, and assigns it to that object. This will ensure that $\mathcal{I}^*(\Phi \boxed{\chi \Rightarrow a}) = 1$. And since $\mathcal{I}^*$ is just like $\mathcal{I}$ with respect to everything other than a — and since we are assured that a was not in $\mathcal{X}$ (the rule requires that it be new) — $\mathcal{I}^*$ will satisfy $\mathcal{X}$ in just the same way that $\mathcal{I}$ did.

This will be clearer with an example. Suppose $\exists x F x$ is part of a satisfiable set of sentences. So assume that $\mathcal{I}$ interprets it. The tree resolution for rule for existentials requires that one take an instance corresponding to a new name. Suppose this is done thus:

$$\exists x F x \; \checkmark a$$
$$|$$
$$F a$$

We cannot simply say that since some model $\mathcal{I}$ satisfies the existential $\exists x F x$, it thereby must also satisfy Fa; plenty of interpretations satisfy the former while falsifying the latter. But what we *can* say, since a is a new name, with no previous commitments specific to it earlier in the tree, is that we can construct a model $\mathcal{I}^*$ that satisfies every sentence above in the tree (in this case, just the one existential), and that also satisfies the new development (in this case, Fa). We do so by assigning the name a to refer to some object that is F in $\mathcal{I}$. We're sure there is such an object there because it is stipulated that $\mathcal{I}$ satisfies the existential.

In general, assuming that $\mathcal{I}$ satisfies every sentence in a branch before the existential rule is performed, there is guaranteed to be a new interpretation, $\mathcal{I}^*$, which is an extension of $\mathcal{I}$ that includes a new name attached to an object in the UD satisfying the existential, which satisfies everything in the branch up to the point after the existential rule is performed. That is to say, like the nine sentential rules considered in Chapter 6, the existential rule can never take you from a satisfiable branch to an unsatisfiable one.

11.1.3 Universals

Suppose a branch of a tree uses the rule for universals:

$$\forall \chi \Phi \setminus a$$
$$|$$
$$\Phi \boxed{\chi \Rightarrow a} \qquad \text{for } any \ a$$

Assume that the set of QL sentences $\mathcal{X}$ above this development is satisfiable. Then some model $\mathcal{I}$ makes it true. The universal rule allows an instance to be developed using any name, so, as before, we cannot guarantee that $\mathcal{I}$ makes the development true, because $\mathcal{I}$ may or may not interpret the name a; but as before, we can be assured if it doesn't, we can extend the interpretation to include it. So consider a new model $\mathcal{I}^*$, which includes the name a. If a wasn't interpreted by $\mathcal{I}$, then it can be assigned to any element of the UD. Since the rest of $\mathcal{I}$ is unchanged, and since $\mathcal{I}(\forall \chi \Phi) = 1$, we know that our new extended interpretation will satisfy $\Phi \boxed{\chi \Rightarrow a}$ too.

That is to say, once again, if we assume that $\mathcal{I}$ satisfies every sentence in a branch before the universal rule is performed, there is guaranteed to be a model — either the very same one, $\mathcal{I}$, or a modification of it, $\mathcal{I}^*$, which assigns a new name to any object in the UD, which satisfies everything in the branch up to the point after the universal rule is performed. In other words, the universal rule can never take you from a satisfiable branch to an unsatisfiable one.

Note that the fact that we can perform this rule multiple times does not interfere with the soundness proof. We have proven that *each* time you perform it, you are guaranteed not to switch from a satisfiable branch to an unsatisfiable one. So no matter how many times you take an instance of a universal, you won't be able to go from a satisfiable set of sentences to an unsatisfiable one.

11.1.4 Negated Existential

The reasoning behind soundness for the negated existential rule is exactly parallel to that for the universal rule. We begin by assuming that some negated universal $\neg\exists\chi\Phi$, is satisfied by $\mathcal{I}$. Here is the rule for negated existentials:

$$\neg\exists\chi\Phi \setminus a$$
$$|$$
$$\neg\Phi\boxed{\chi \Rightarrow a} \qquad \text{for } \textit{any } a$$

We want to prove that the result of this rule is also satisfied, either by $\mathcal{I}$ itself (if a was interpreted in $\mathcal{I}$), or by an extension of it $\mathcal{I}^*$, that preserves the satisfaction of everything above (if a was a new name). Since $\mathcal{I}$ satisfies $\neg\exists\chi\Phi$, it makes every substitution instance for χ of Φ false. If a was interpreted by $\mathcal{I}$ already, then $\mathcal{I}(\Phi\boxed{\chi \Rightarrow a}) = 0$. If it wasn't, the new model $\mathcal{I}^*$ will assign the new name to some object in the UD of the original model; since no object in that model satisfied Φ, $\mathcal{I}^*(\Phi\boxed{\chi \Rightarrow a}) = 0$. Either way, our interpretation falsifies $\Phi\boxed{\chi \Rightarrow a}$, and so satisfies that sentence's negation, which is the continuation of the branch.

So this rule too can never take us from a satisfiable set of QL sentences to an unsatisfiable one.

11.1.5 Negated Universal

Negated universals are similar to existentials. Assume that a negated universal is part of a set of sentences satisfied by $\mathcal{I}$, and that this rule is then applied:

$$\neg\forall\chi\Phi \checkmark a$$
$$|$$
$$\neg\Phi\boxed{\chi \Rightarrow a} \qquad \text{where } a \text{ is } \textit{new}$$

Construct a new interpretation $\mathcal{I}^*$, which differs from $\mathcal{I}$ only in that it includes an interpretation of the new name a, and assigns that name to some object that

falsifies Φ. We know there is at least one such object because we are assuming that $\mathcal{I}$ satisfies the negated universal. Then our new interpretation $\mathcal{I}^*$ satisfies the new development of the branch. It also satisfies everything above the branch, just like $\mathcal{I}$ did, because nothing above the branch included the name a.

That last bit of reasoning relied centrally on the requirement that we're taking a new name. We saw in the introduction to this chapter that if we do not include that requirement, soundness would be violated.

11.1.6 Summarizing the soundness proof

We have now shown, for our four quantifier rules, that each of them has the following property: you can never start with a branch that is satisfiable, and use that rule to extend that branch into one that is unsatisfiable. Since we've also shown that the nine sentential rules also have this property, we've effectively shown that there is no possible way to start with a satisfiable set of sentences and develop the branch into one that is not satisfiable. This in turn means that if the branch starts with a satisfiable set of sentences, the branch will never close. But that's just what soundness says: if the root is satisfiable, the tree is guaranteed to remain open. Soundness is proven.

11.2 Completeness

Completeness says that if a branch of a completed tree remains open, then the root is satisfiable. We prove this by assuming that we have an open completed branch, and use it to construct an interpretation that satisfies every sentence in that branch, which includes the root. The proof for completeness of our QL tree system is structurally just like the one given in Chapter 6.

Given a completed open branch, we construct a model $\mathcal{I}$, based on that branch, as follows: for any predicate $\mathcal{F}$, if some QL atom $\mathcal{F}a_1, \ldots, a_n$ is in the branch — i.e., if P or Fa or Rab is in the branch — then $\mathcal{I}$ makes that atom true, by putting $<a_1, \ldots, a_n>$ in the extension of $\mathcal{F}$. And if $\neg\mathcal{F}a_1, \ldots, a_n$ is in the branch, $\mathcal{I}$ excludes $<a_1, \ldots, a_n>$ from the extension of $\mathcal{F}$, thus making the negation of the atom true. This is of course just the way that we construct interpretations from open branches of completed trees.

Now we will prove, for every sentence in QL, that if it is in the open branch, it is made true by $\mathcal{I}$. The QL atoms trivially meet this criterion — $\mathcal{I}$ was designed precisely to satisfy them. We will prove by induction that every possible QL sentence also meets this criterion. In §6.6 we showed, for each propositional connective, if you construct a more complex SL sentence out of simpler SL

sentences that have this criterion, the more complex one will too. That proof carries on unchanged here. So it remains only to show that the same is true of our four quantifier rules.

11.2.1 Existential

Consider an existential — a QL sentence of the form $\exists\chi\Phi$. We need to prove that if it is in the open, completed branch, $\mathcal{I}$ satisfies it. Since the branch is complete, we know that the existential rule has been performed to resolve this sentence. So the branch includes a substitution instance of Φ that used a new name. For our present purposes, it doesn't actually matter whether the name was new — the fact that there is some instance of Φ in the branch already is enough to prove what we need to prove. Since there is an instance of Φ in the branch, if it is satisfied by $\mathcal{I}$, the existential $\exists\chi\Phi$ must be satisfied by $\mathcal{I}$ too.

So, just as we showed for the nine sentential rules, the existential rule has this important property: in a completed tree, any interpretation that satisfies the simpler sentences below the existential development, must also satisfy the existential above it.

11.2.2 Universal

Suppose a universal sentence $\forall\chi\Phi$ appears in a completed open branch. Since the branch is complete, that means that, for every name a in the branch, $\Phi\boxed{\chi \Rightarrow a}$ is also in the branch. We therefore assume that $\mathcal{I}$ satisfies each $\Phi\boxed{\chi \Rightarrow a}$; so $\mathcal{I}$ must also satisfy $\forall\chi\Phi$. Because the UD for $\mathcal{I}$ includes only those names that occur in the branch, every instance of Φ is included, so the universal is true.

Once again, any interpretation that satisfies everything below the universal development, must also satisfy the universal above it.

11.2.3 Negated Existential

Negated existentials work just like universals. If $\neg\exists\chi\Phi$ is in a completed open branch, then for every name a in $\mathcal{I}$, $\neg\Phi\boxed{\chi \Rightarrow a}$ is below it in the branch. And if $\mathcal{I}$ satisfies each of these negations, it will also satisfy the negated existential.

11.2.4 Negated Universal

Negated universals work just like existentials. If $\neg\forall\chi\Phi$ is in the branch, then some instance of the negation $\neg\Phi$ is in the branch below. If $\mathcal{I}$ satisfies some instance of $\neg\Phi$, then, given the definition of truth for negation and universals in QL, it will also satisfy $\neg\forall\chi\Phi$.

11.2.5 Summarizing the completeness proof

The sentence shapes just considered, combined with the nine shapes considered in §6.6, correspond to all the possible QL sentences. So we have proven that, for any possible QL sentence Φ, if an interpretation satisfies the simpler sentences below it in the branch, that interpretation also satisfies Φ itself. Since we also have a recipe for constructing an interpretation $\mathcal{I}$ that is guaranteed to satisfy the atoms, we can prove by induction that it can satisfy everything in the branch, including the root. A completed open branch guarantees a satisfiable root. Completeness is proven.

Practice Exercises

⋆ **Part A** Following are possible modifications to our QL tree system. For each, imagine a system that is like the system laid out in this chapter, except for the indicated change. Would the modified tree system be sound? If so, explain how the proof given in this chapter would extend to a system with this rule; if not, give a tree that is a counterexample to the soundness of the modified system.

1. Change the rule for existentials to this rule:

$$\exists\chi\Phi \ \checkmark a$$
$$|$$
$$\Phi\boxed{\chi \Rightarrow a} \qquad \text{for } any \ a$$

2. Change the rule for existentials to this rule:

$$\exists\chi\Phi \ \checkmark d$$
$$|$$
$$\Phi\boxed{\chi \Rightarrow d} \qquad \text{(whether or not } d \text{ is new)}$$

3. Change the rule for existentials to this rule:

$\exists \chi \Phi \checkmark$
|
$\Phi \boxed{\chi \Rightarrow a}$ for 3 different names, old or new

$\Phi \boxed{\chi \Rightarrow b}$

$\Phi \boxed{\chi \Rightarrow c}$

4. Change the rule for universals to this rule:

$\forall \chi \Phi \checkmark$
|
$\Phi \boxed{\chi \Rightarrow a}$ for 3 different names, old or new

$\Phi \boxed{\chi \Rightarrow b}$

$\Phi \boxed{\chi \Rightarrow c}$

5. Change the rule for existentials to this rule:

$\exists \chi \Phi \checkmark$
|
$\Phi \boxed{\chi \Rightarrow a}$ for 3 new names

$\Phi \boxed{\chi \Rightarrow b}$

$\Phi \boxed{\chi \Rightarrow c}$

6. Change the rule for universals to this rule:

$\forall \chi \Phi \checkmark a$
|
$\Phi \boxed{\chi \Rightarrow a}$ where a is *new*

7. Change the rule for conjunction to this rule:

$$\Phi \,\&\, \Psi \;\checkmark$$
$$|$$
$$\exists\chi\Phi \qquad \text{where } \chi \text{ does not occur in } \Phi$$
$$\Psi$$
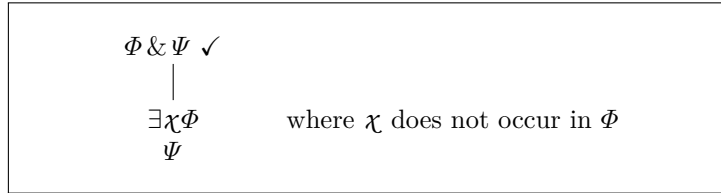
8. Change this requirement (given on page 191)...

> A branch is COMPLETE if and only if either (i) it is closed, or (ii) every resolvable sentence in every branch has been resolved, and for every general sentence and every name *a* in the branch, the *a* instance of that general sentence has been taken.

...to this one:

> A branch is COMPLETE if and only if either (i) it is closed, or (ii) every resolvable sentence in every branch has been resolved, and for every general sentence, *at least one instance of* that general sentence has been taken.

9. Change the branch completion requirement to:

> ...and for every general sentence and every name *a that is above that general sentence in the branch*, the *a* instance of that general sentence has been taken.

10. Change the branch completion requirement to:

> ...and for every general sentence and every name *a* in the branch, the *a* instance of that general sentence has been taken, *and at least one additional new instance of that general sentence has also been taken.*

$\star$ **Part B** For each of the rule modifications given in Part A, would the modified tree system be complete? If so, explain how the proof given in this chapter would extend to a system with this rule; if not, give a tree that is a counterexample to the completeness of the modified system.

Chapter 12

Identity

This chapter extends QL by adding logical resources for discussion of *identity*. In the logical sense, identity is about which object something is. Two things aren't 'identical' in our sense just because they look exactly alike. (Identical twins aren't *logically* identical.) Neither does identity, in our logical sense, have to do with the way an individual conceives of oneself. (We're not getting into whether one *identifies as* a woman, etc.) Instead, we will develop logical vocabulary to talk about whether, e.g., the person who robbed the bank is *the same person as* the person who shot Uncle Ben.

It is already possible to talk about identity in QL, by assigning identity and related concepts to predicates in interpretation keys. (In this sense, it's possible to talk about pretty much anything in QL.) However, there are a few reasons why we may wish to have a more robust logical notion of identity than that. We begin by examining some of these motivations. Identity will strengthen QL, increasing its expressive power in many ways.

12.1 Motivating identity as a logical category

Consider this sentence:

1. Only you can prevent forest fires.

How should we translate it into QL? Here is one simple option. We could use this symbolization key:

UD: people
Px: only x can prevent forest fires
u: you

And then we could translate sentence 1 straightforwardly as Pu. This might be adequate for some purposes, but this translation misses out on a key fact about sentence 1. Notice that sentence 1 is inconsistent with sentence 2:

2. Only your mother can prevent forest fires.

If we add a name for your mother to the symbolization key above,

m: your mother

we can translate sentence 2 as Pm. The problem is that Pu and Pm are obviously consistent in QL, as this simple model demonstrates:

$$\text{UD} = \{u,\, m\}$$
$$\text{extension}(P) = \{u,\, m\}$$

The 'only' in each sentence has just been made a part of the predicate, in a way that ignores its logical force. (Compare the analogous discussion in the Introduction to Chapter 8 that motivated extending SL with predicates and quantifiers.) If *only* you can prevent forest fires, that means that no one *else* — i.e., no one who *is not* you — can prevent forest fires. We don't yet have the logical vocabulary to express these thoughts in a strong enough way.

These sentences give rise to a related issue:

3. Peter Parker lives in Queens.
4. Peter Parker is Spider-Man.
5. Spider-Man lives in Queens.

Here is a natural interpretation key for these sentences, given the resources we've seen so far.

UD: people and places
Lxy: x lives in y
Ixy: x is y
p: Peter Parker
s: Spider-Man
q: Queens

Then sentences 3–5 can be translated thus:

    3. *Lpq*
    4. *Ips*
    5. *Lsq*

But the argument from the English 3 and 4 to 5 seems valid, whereas the corresponding argument form is obviously invalid in QL. This model satisfies the premises and falsifies the conclusion:

$$\text{UD} = \{\text{p, s, q}\}$$

extension(L) =

Lxy	p	s	q
p	-	-	1
s	-	-	0
q	-	-	-

extension(I) =

Ixy	p	s	q
p	-	1	-
s	-	-	-
q	-	-	-

Given the interpretation key above, this is a model where Peter is Spider-Man, but where Peter and Spider-Man do not live in the same place. This does not correspond to a genuine possibility; if Peter and Spider-Man are the same person, then they must live in the same place. This is an important logical fact about identity. But in the formalism given, I is just another predicate like L, susceptible to many different interpretations and extensions.

Here is a third example.

    6. Mozart composed *Le Nozze di Figaro*.
    7. Mozart composed *Don Giovanni*.
    8. Mozart composed at least two things.

We introduce the obvious interpretation key:

  UD: people and operas
  Cxy: *x* composed *y*
    m: Mozart
    n: *Le Nozze di Figaro*
    g: *Don Giovanni*

Then we might translate the argument thus:

$$Cmn$$
$$Cmg$$
$$\therefore \; \exists x Cmx \,\&\, \exists y Cmy$$

This is a valid argument form. But the problem is that $\exists x Cmx \,\&\, \exists y Cmy$ is a very bad translation of 'Mozart composed at least two things'. Notice that it is the conjunction of two existentials, each of which says that Mozart composed at least one thing. Suppose he composed only one thing; then each conjunct is true, so the conjunction is true. To say that he composed at least *two* things, it's not enough to say that he composed at least one thing, and then to say again that he composed at least one thing — we need some way to ensure that the x and y are *not the same thing*.

What all three of these cases — the forest fire case, the Spider-Man case, and the Mozart case — show us is that if we want to reflect the logical relationships having to do with identity, we need special logical vocabulary to do so. Just as we introduced the '$\forall$' and '$\exists$' formalisms to reflect logical facts about quantifiers, we also need a special symbol for identity. That symbol will be '$=$'. Like our other logical symbols — '$\supset$', '$\neg$', '$\forall$', etc. — its meaning will be fixed as part of our language. Interpretation keys tell us what predicates like Fx mean; they don't get to specify the meaning of the logical terms.

When we introduced quantifiers and predicates, we described the shift as switching to a new language, from SL to QL. In fact, however, QL isn't so much a change away from SL as it is an addition to it. (All SL sentences are QL sentences, and all SL entailments are also QL entailments.) Here we will make one more addition, corresponding to identity. We won't introduce a whole new name for our extended version of QL; it can be specified descriptively as 'quantified logic with identity'. (The choice about whether to introduce a new language name is a purely conventional one; we're here just following ordinary practice.)

12.2 $=$

We revise the definitions of grammaticality and truth in QL to add the following stipulation:

For any two names $\boldsymbol{a}$ and $\boldsymbol{b}$, '$\boldsymbol{a}=\boldsymbol{b}$' is an atomic sentence in QL. It is true in a model just in case, in that model, the referent of $\boldsymbol{a}$ is the same object as the referent of $\boldsymbol{b}$; i.e., the model assigns $\boldsymbol{a}$ and $\boldsymbol{b}$ to the same object. For example, here is a model:

$$UD = \{\text{Peter Parker, Aunt May}\}$$
$$\text{referent}(p) = \text{Peter Parker}$$
$$\text{referent}(s) = \text{Peter Parker}$$
$$\text{referent}(m) = \text{Aunt May}$$

In this model, $p=s$ is true, and $p=m$ and $s=m$ are false. We'll introduce a familiar shorthand for negating identity claims; $s{\neq}m$ is our preferred abbreviation for $\neg s=m$. (Remember that '$s=m$' is an atomic sentence, so '$\neg s=m$' is the negation of that sentence. It is tempting, when one looks at that sentence, to attach the negation only to s, so that the sentence says that the 'negation of s' is identical to m. But this way of reading it ultimately makes no sense; names do not have negations in QL. Only sentences do. This is part of the reason we will usually prefer to write $s{\neq}m$, rather than the potentially confusing $\neg s=m$. But if you do see or write the latter, remember that the negation applies to the whole identity claim.)

A logical notion of identity is just what is needed to overcome the shortcomings of the translations in the introduction to this chapter.

12.3 Identity and 'no one else'

Consider again sentence 1:

1. Only you can prevent forest fires.

In the introduction we tried translating this into QL with a predicate 'only _____can prevent forest fires'; now we can reflect its logical structure more fully. So we'll let our predicate just indicate the ability to prevent forest fires:

UD: people
Px: x can prevent forest fires
u: you

If only you can prevent forest fires, that means that you can prevent forest fires, and, moreover, no one *else* — no one who is not you — can prevent forest fires. The first part is simple. You can prevent forest fires: Pu. Now we must say that no one else can prevent forest fires. Another way to say that is this: it is *not* the case that there is someone *who is not you* who can prevent forest fires. So you can think of that as a negated existential: $\neg\exists x(x{\neq}u \mathbin{\&} Px)$. Putting these two pieces together as a conjunction, we can translate sentence 1 as:

1. $Pu \mathbin{\&} \neg\exists x(x{\neq}u \mathbin{\&} Px)$

A second way of characterizing the 'no one else' clause would translate it as a universal claim, saying, of everything, that if it is not you, then it cannot prevent forest fires: $\forall x(x{\neq}u \supset \neg Px)$. This is equivalent to $\neg\exists x(x{\neq}u \,\&\, Px)$; either is an acceptable translation. (By the end of this chapter we'll be able to prove this equivalence.)

Using similar reasoning, one can translate other kinds of sentences that talk about objects *other than* ones just mentioned. For example, suppose we want to say that Rebecca only thinks about herself. She thinks about herself, and there is no one else that she thinks about. Using r for Rebecca and Txy for 'x thinks about y', we could say:

$$Trr \,\&\, \neg\exists x(x{\neq}r \,\&\, Trx)$$

There are more examples of similar kinds of translation exercises in the practice problems at the end of this chapter.

12.4 Identical objects satisfy identical predicates

In §12.1, we saw that treating identity as a simple predicate prevented the validity of intuitively valid arguments like this one:

3 Peter Parker lives in Queens.
4 Peter Parker is Spider-Man.
5 Spider-Man lives in Queens.

The problem was that if the extension of the identity predicate can simply be presented in a model, without regard for the other features of the model, there was no guarantee that p and s — the names for Peter Parker and Spider-Man — were related in any particular way, just because <Peter Parker, Spider-Man> happened to be assigned in the extension of the I predicate.

But if we translate the identity predicate with the QL identity claim $p{=}s$, this requires that the names s and p be assigned to the *very same object* in the model. Consequently, for any sentence Φ involving the name p, $\Phi \boxed{p \Rightarrow s}$ — the sentence that results from replacing each p in Φ with an s — will have the same truth value as Φ. If Fp is true, then the object named by p is in the extension of F. And if $p{=}s$, then the object named by p is *the very same object as* the object named by s. So of course the object named by s (that same object) must also be in the extension of F. In general, identical objects have identical properties.

(The converse is not true in general. There is no rule in QL that says that if all the same wffs are true of two different names, they must be two names for the same object.)

If Peter is Spider-Man, then everything true of Peter is also true of Spider-Man. So this argument form, unlike the first translation we attempted, is valid in QL:

> *Lpq*
> *p=s*
> ∴ *Lsq*

In §12.7 below we'll extend our tree method to claims with identity, which will give us a formal means for proving the validity of arguments like this one. (The natural deduction system for QL given in Chapter 13 will give us another.)

12.5 Quantity

Identity also lets us talk about quantity in a way that wasn't available before. In §12.1 we observed some challenges with translating claims about the number of objects that satisfy a particular description. For example, we struggled with sentences like 'Mozart composed more than one opera'. But identity makes such claims straightforward.

UD:	people and operas
Cxy:	*x* composed *y*
Ox:	*x* is an opera
m:	Mozart
n:	*Le Nozze di Figaro*
g:	*Don Giovanni*

We want to say that there are two things, that have the following features: they are different from each other, they are both operas, and they are both composed by Mozart. A double-existential sentence, listing those conjuncts, will suffice:

$$\exists x \exists y (x \neq y \,\&\, Ox \,\&\, Oy \,\&\, Cmx \,\&\, Cmy)$$

Strictly speaking, QL conjunctions only have two conjuncts, so to write this sentence scrupulously, we'd need to include brackets to signify that we're talking about conjunctions of conjunctions:

$$\exists x \exists y (x \neq y \,\&\, [(Ox \,\&\, Oy) \,\&\, (Cmx \,\&\, Cmy)])$$

In sentences like these — and the longer ones listing more conditions that will feature soon — it is usually preferable for readability to leave these internal brackets out. We will write as if our system allows arbitrary numbers of conjuncts, with the understanding that, should the need arise, we could always put them back in explicitly. (We'll rarely have occasion to do so.)

This sentence says that Mozart wrote at least two operas. To say that he wrote at least *three* operas would require another quantifier with a new variable z, and the specification that z be an opera, written by Mozart, and non-identical to both of the ones previously mentioned:

$$\exists x \exists y \exists z (x{\neq}y \,\&\, x{\neq}z \,\&\, y{\neq}z \,\&\, Ox \,\&\, Oy \,\&\, Oz \,\&\, Cmx \,\&\, Cmy \,\&\, Cmz)$$

As you can see, these sentences get long and complicated rather quickly.

Suppose we wanted to say that Mozart composed *exactly* two operas — no more, and no fewer. There are two reasonably natural choices for how to say that. One is to conjoin the first QL sentence of this section — Mozart composed at least two operas — with the negation of the second — Mozart did not compose at least three operas. The result is:

$$\exists x \exists y (x{\neq}y \,\&\, Ox \,\&\, Oy \,\&\, Cmx \,\&\, Cmy) \,\&\,$$
$$\neg\exists x \exists y \exists z (x{\neq}y \,\&\, x{\neq}z \,\&\, y{\neq}z \,\&\, Ox \,\&\, Oy \,\&\, Oz \,\&\, Cmx \,\&\, Cmy \,\&\, Cmz)$$

A slightly more efficient way of saying the same thing, rather than conjoining a new negated triple existential, would be to add, within the scope of the first two quantifiers, a conjunct to the effect that every opera composed by Mozart is one of the two mentioned in that first conjunction. To say that any opera composed by Mozart is either x or y is to say that, for any z, if z is an opera composed by Mozart, then it is either x or y. In other words: $\forall z[(Oz \,\&\, Cmz) \supset (z{=}x \lor z{=}y)]$. So another way to say that Mozart composed exactly two operas is:

$$\exists x \exists y ((x{\neq}y \,\&\, Ox \,\&\, Oy \,\&\, Cmx \,\&\, Cmy) \,\&\, \forall z[(Oz \,\&\, Cmz) \supset (z{=}x \lor z{=}y)])$$

So identity gives us a way to talk about quantities, albeit not a particularly efficient one.

12.6 Definite descriptions

In 1905, Bertrand Russell famously characterized *definite descriptions* in terms of identity. A definite description is a description that implies that only one object satisfies it. Paradigmatically, definite descriptions are ones that use the definite article 'the'. If I say 'the baby is hungry', I'm saying that the *one and only baby we might be talking about* is hungry. Russell was motivated in part by the apparent fact that one can use this sort of language in a meaningful way even if one is wrong about whether there's any baby around. If there is no baby — the crying I'm hearing is a recording — my statement is false, but it's still

meaningful. For this reason, Russell was reluctant to suppose that we should understand 'the baby' as a name. Remember, in QL, all names have to refer to objects in the UD. Instead, my sentence can be understood as an existentially quantified claim about a unique baby. If I say 'the baby is hungry', according to Russell, I'm in effect saying three things: there is a baby, there's no other baby than that one, and that baby is hungry. That second element, the uniqueness claim, can be expressed in QL with identity.

 UD: People in this house
  Bx: x is a baby.
  Hx: x is hungry.
   j: Jonathan

A sentence like 'Jonathan is hungry' is straightforwardly translated as Hj. According to Russell's theory of definite descriptions, 'the baby is hungry' has a much more complex logical form: there is a baby x, there is no baby other than x, and x is hungry.

$$\exists x[Bx \,\&\, \neg\exists y(By \,\&\, y{\neq}x) \,\&\, Hx]$$

One of the interesting features of Russell's theory is that 'the baby is not hungry' is not the negation of 'the baby is hungry'. Instead, the negation applies only to the hunger predication in the last conjunct:

$$\exists x[Bx \,\&\, \neg\exists y(By \,\&\, y{\neq}x) \,\&\, \neg Hx]$$

The reason Russell designed his theory this way was that he thought that both of these sentences equally implied that there is a baby. If there is no baby, then you'd be mistaken in saying either 'the baby is hungry' or 'the baby is not hungry'. Consequently, one can't be the negation of the other. As a treatment of the truth conditions of English sentences, Russell's theory is controversial. Some philosophers of language think that sentences that seem to presuppose the existence of something that isn't there aren't straightforwardly false, but are rather defective in some other way — perhaps they fail to be meaningful at all, or perhaps they take on some truth value other than true or false. These matters are beyond our present scope. We will remain neutral on whether Russell's theory is an accurate treatment of English; it is relevant for us because it provides an interesting and useful case for translation of sentences into QL with identity.

The general statement of Russell's theory of definite descriptions is, if you have a sentence applying a predicate $\mathcal{P}$ to the referent of a definite description $\mathcal{D}$, translate it thus: there is something χ that is $\mathcal{D}$, there is nothing other than χ that is $\mathcal{D}$, and χ is $\mathcal{P}$.

12.7 Identity and trees

Because we have extended QL to include claims of identity, we also need to modify our tree system. Notice that if we do not do so, the system will not be complete. For example, the rules offered in Chapter 11 do not provide a way to demonstrate that $Fa, \neg Fb, a{=}b \models \bot$, even though those sentences are obviously inconsistent. Since each of those sentences are atoms or negated atoms in QL, there is nothing to resolve, and there is no contradiction to close the tree. We need to add new rules to deal with identity.

Here is an obvious one:

> If a branch contains an identity claim $a{=}b$ and some sentence Φ, then you may add $\Phi \boxed{a \Rightarrow b}$ or $\Phi \boxed{b \Rightarrow a}$ to the branch.

Recall that '$\Phi \boxed{a \Rightarrow b}$' just means the sentence you get by replacing every 'a' in Φ with 'b'. With this rule, we can easily demonstrate that the set of sentences mentioned above closes:

$\{Fa, \neg Fb, a{=}b\} \vdash \bot$

1. Fa
2. $\neg Fb$
3. $a{=}b$
        |
4. Fb $1, 3 =$
        $\times$
        2, 4

We will add this rule to our tree system. Notice that it is not a *resolution* rule — we do not put a check mark next to either sentence referenced. They may be used again.

Since our tree system is supposed to be *complete*, we need to *ensure* that the tree will close; it's not enough that we have a rule that allows that it might. So, as in the case of the general rules discussed in Chapter 10, we will also add an additional requirement governing branch completion. What we need is to make sure that we've made enough identity substitutions to demonstrate everything that needs demonstrating. The way we'll do this is by requiring, for every identity claim in the branch, that, for at least one of its directions (left-to-right or right-to-left) the substitution has been performed in that direction on every atomic sentence or negated atomic sentence in the branch.

More precisely stated, an open branch is complete only if, for any identity claim

$a=b$ in the branch, either, for any atomic sentence or negated atomic sentence Φ containing a in the branch, $\Phi\boxed{a \Rightarrow b}$ is in the branch, or, for any atomic sentence or negated atomic sentence Φ containing b in the branch, $\Phi\boxed{b \Rightarrow a}$ is in the branch.

So, adding this requirement to the definition of branch completion given on page 191, we get:

A branch is COMPLETE if and only if either (i) it is closed, or (ii) all of the following conditions are met:

1. Every resolvable sentence in the branch has been resolved;

2. For every general quantified sentence Φ and every name a in the branch, the a instance of Φ is in the branch; and

3. For every identity claim $a=b$ in the branch, either:

    (a) for every atomic sentence or negated atomic sentence Φ containing a in the branch, $\Phi\boxed{a \Rightarrow b}$ is in the branch, or

    (b) for every atomic sentence or negated atomic sentence Φ containing b in the branch, $\Phi\boxed{b \Rightarrow a}$ is in the branch.

We must also add one more rule to our tree system. Previously we said that branches only close when they contain some sentence along with its negation. We add one more condition that suffices for branch closure. If any branch contains a sentence $a \neq a$, then that branch closes. In no model can something be non-identical with itself.

Adding this to the rule given on page 190:

A branch closes if and only if either (i) it contains some sentence Φ along with its negation, $\neg\Phi$, or (ii) it contains, for some name a, the sentence $a \neq a$.

We need this rule, for example, to prove with a tree that $\forall x\, a=x$ is inconsistent with $\exists x\, x \neq a$:

$\{\forall x\, a{=}x, \exists x\, x{\neq}a\} \vdash \bot$

1. $\forall x\, a{=}x$ \b
2. $\exists x\, x{\neq}a$ ✓b
          |
3. $b{\neq}a$ 2 ∃
          |
4. $a{=}b$ 1 ∀
          |
5. $b{\neq}b$ 3, 4 =
          ×
          5

We must also take care, when constructing models from completed open branches, to respect identity claims. If for example the sentence $b{=}c$ appears in an open branch, we will not construct a model with two different objects, b and c — instead, we'll just pick one of them as the object indicated, and assign both names to it. For example, consider this tree:

1. $\forall x(Rax \supset x{=}b)$ \a, b, c
2. $\exists x Rax$ ✓c
             |
3. Rac 2 ∃
             |
4. $Raa \supset a{=}b$ ✓   1 ∀
5. $Rab \supset b{=}b$ ✓   1 ∀
6. $Rac \supset c{=}b$ ✓   1 ∀

7. $\neg Rac$ $c{=}b$ 6 ⊃
       × |
      3, 7
8. Rab 3, 7 =

9. $\neg Rab$ $b{=}b$ 5 ⊃
        × |
      8, 9
10. $\neg Raa$ $a{=}b$ 4 ⊃
         ↑ |
11. Rbb 8, 10 =
              |
12. Rbc 3, 10 =
             ↑

We end up with two open branches. We only need one to demonstrate that the root is satisfiable, but it is useful practice to consider both. Let's begin with

the shorter open branch to the left, that completes at line 10. Examining that branch, we see three names, a, b, and c, and three atomic sentences concerning the extension of R: Rac, Rab, and $\neg Raa$. Using our previous method for constructing models, we would have posited a three-object UD and related those objects via R in the appropriate way. Now, however, we also have identity claims to consider. There are two identity claims in the branch. $b{=}b$ doesn't tell us anything interesting — of *course* that object is identical with itself — but the presence of $c{=}b$ indicates that our model will not include separate objects b and c. We'll only posit one object in the UD for those two names, and another object for a. We can pick either letter for the object; let's call it b. (Remember, it doesn't matter what kinds of objects we pick for the UD; it could be Reed Richards for all it matters. We typically pick letters because they are easy to keep track of.)

So the model suggested by the first open branch here is:

$$\text{UD} = \{\text{a, b}\}$$
$$\text{referent}(a) = \text{a}$$
$$\text{referent}(b) = \text{b}$$
$$\text{referent}(c) = \text{b}$$

$$\text{extension}(R) = \begin{array}{c|cc} Rxy & a & b \\ \hline a & 0 & 1 \\ b & - & - \end{array}$$

The longer open branch includes additional identity claims. It includes $a{=}b$ and $c{=}b$, which of course implies that all three of these names are names for the same object. So the UD for the model corresponding to the right branch will have only one object, which all three names denote.

$$\text{UD} = \{\text{a}\}$$
$$\text{referent}(a) = \text{a}$$
$$\text{referent}(b) = \text{a}$$
$$\text{referent}(c) = \text{a}$$

$$\text{extension}(R) = \begin{array}{c|c} Rxy & a \\ \hline a & 1 \end{array}$$

It it sometimes helpful, for an intuitive grip on what is going on in these trees, to think about an English translation alongside the formalisms. For example, suppose that Rxy is interpreted as 'x loves y'. Line 1 of the tree says that person a loves at most one person; line 2 says that a loves at least one person. The tree works out two ways that might be: the one and only person a loves could be a, or it could be someone else.

Practice Exercises

⋆ **Part A** Using the symbolization key given, translate each English-language sentence into QL with identity. For sentences containing definite descriptions, assume Russell's theory.

   UD: people
   Kx: x knows the combination to the safe.
   Sx: x is a spy.
   Vx: x is a vegetarian.
  Txy: x trusts y.
      h: Hofthor
      i: Ingmar

1. Hofthor is a spy, but no vegetarian is a spy.
2. No one knows the combination to the safe unless Ingmar does.
3. No spy knows the combination to the safe.
4. Neither Hofthor nor Ingmar is a vegetarian.
5. Hofthor trusts a vegetarian.
6. Everyone who trusts Ingmar trusts a vegetarian.
7. Everyone who trusts Ingmar trusts someone who trusts a vegetarian.
8. Only Ingmar knows the combination to the safe.
9. Ingmar trusts Hofthor, but no one else.
10. The person who knows the combination to the safe is a vegetarian.
11. The person who knows the combination to the safe is not a spy.

⋆ **Part B** Using the symbolization key given, translate each English-language sentence into QL with identity. For sentences containing definite descriptions, assume Russell's theory.

   UD: cards in a standard deck
   Bx: x is black.
   Cx: x is a club.
   Dx: x is a deuce.
   Jx: x is a jack.
   Mx: x is a man with an axe.
   Ox: x is one-eyed.
   Wx: x is wild.

1. All clubs are black cards.
2. There are no wild cards.
3. There are at least two clubs.
4. There is more than one one-eyed jack.

5. There are at most two one-eyed jacks.
6. There are exactly two black jacks.
7. There are exactly four deuces.
8. The deuce of clubs is a black card.
9. One-eyed jacks and the man with the axe are wild.
10. If the deuce of clubs is wild, then there is exactly one wild card.
11. The man with the axe is not a jack.
12. The deuce of clubs is not the man with the axe.

⋆ **Part C** Using the symbolization key given, translate each English-language sentence into QL with identity.

UD: people, generations, and monsters
Gx: x is a generation.
Hx: x is human.
Sx: x is a slayer.
Vx: x is a vampire.
Dx: x is a demon.
Wx: x is a werewolf.
Fx: x is a force of darkness.
Axy: x will stand against y.
Bxy: x is born in generation y.
Kxy: x will kick y.
b: Buffy
f: Faith
w: Willow

1. Buffy and Willow were born unto the same generation.
2. There is no more than one slayer born in each generation.
3. A slayer other than Buffy is one of the forces of darkness.
4. Willow will stand against any force of darkness other than a werewolf.
5. Faith will kick everyone except herself.
6. Buffy will kick anyone who stands against a slayer, unless they are also kicking vampires or demons.
7. In every generation a slayer is born.
8. In every generation a slayer is born. She will stand against the vampires, demons, and forces of darkness.
9. In every generation a slayer is born. She alone will stand against the vampires, demons, and forces of darkness.

Part D Using the symbolization key given, translate each English-language sentence into QL with identity. For sentences containing definite descriptions, assume Russell's theory.

UD: animals in the world

Bx: x is in Farmer Brown's field.
Hx: x is a horse.
Px: x is a Pegasus.
Wx: x has wings.

1. There are at least three horses in the world.
2. There are at least three animals in the world.
3. There is more than one horse in Farmer Brown's field.
4. There are three horses in Farmer Brown's field.
5. There is a single winged creature in Farmer Brown's field; any other creatures in the field must be wingless.
6. The Pegasus is a winged horse.
7. The animal in Farmer Brown's field is not a horse.
8. The horse in Farmer Brown's field does not have wings.

⋆ **Part E** Construct a model to demonstrate each of the following. If you wish, you can draw a tree to help you answer these questions; however, it is good conceptual practice to tackle some of these questions directly by thinking about just what you'd need to put in your model.

1. Show that $\{\neg Raa, \forall x(x{=}a \lor Rxa)\}$ is consistent.
2. Show that $\{\forall x\forall y\forall z(x{=}y \lor y{=}z \lor x{=}z), \exists x\exists y\ x{\neq}y\}$ is consistent.
3. Show that $\{\forall x\forall y\ x{=}y, \exists x\ x{\neq}a\}$ is inconsistent.
4. Show that $\exists x(x{=}h\,\&\,x{=}i)$ is contingent.
5. Show that $\{\exists x\exists y(Zx\,\&\,Zy\,\&\,x{=}y), \neg Zd, d{=}s\}$ is consistent.
6. Show that '$\forall x(Dx \supset \exists yTyx) \therefore \exists y\exists z\ y{\neq}z$' is invalid.

⋆ **Part F** Construct a tree to test the following entailment claims. If they are false, provide a model that demonstrates this.

1. $\models \forall x\forall y(x{=}y \supset y{=}x)$
2. $\models \forall x\exists y\ x{=}y$
3. $\models \exists x\forall y\ x{=}y$
4. $\exists x\forall y\ x{=}y \models \forall x\forall y(Rxy \equiv Ryx)$
5. $\models \neg\forall x\forall y\forall z[(Axy\,\&\,Azx\,\&\,y{=}z) \supset Axx]$
6. $\forall x\forall y\ x{=}y \models \exists xFx \equiv \forall xFx$
7. $\forall x(x{=}a \lor x{=}b), Ga \equiv \neg Gb \models \neg\exists x\exists y\exists z(Gx\,\&\,Gy\,\&\,\neg Gz)$
8. $\forall x(Fx \supset x{=}f), \exists x(Fx \lor \forall y\ y{=}x) \models Ff$
9. $\exists x\exists yDxy \models \forall x_1\forall x_2\forall x_3\forall x_4[(Dx_1x_2\,\&\,Dx_3x_4) \supset (x_2{\neq}x_3 \lor Dx_1x_4)]$

Part G In §12.5 we looked at two different translations of 'Mozart wrote exactly two operas'. Use trees to prove that they are equivalent.

Part H Translate these arguments into QL with identity, and evaluate them for validity with a tree. (Don't be surprised or discouraged if some of these trees end up very complex.)

1. Dudley will threaten anyone who threatens anyone. Therefore, Dudley will threaten himself.
2. The exam is easy. Therefore every exam Sheila took was easy. (Use Russell's theory of definite descriptions.)
3. Three wise men visited Jesus. Every wise man who visited Jesus gave Jesus a gift. Therefore, Jesus received more than one gift.
4. Worf is the only Klingon in Starfleet. Everyone in Starfleet is brave. All brave Klingons are warriors. Therefore, there is at least one brave warrior in Starfleet.
5. Worf is the only Klingon in Starfleet. Everyone in Starfleet is brave. All brave Klingons are warriors. Therefore, there is exactly one brave warrior in Starfleet.
6. Every person likes every kind of sandwich that is tasty. Jack is a person. Jack likes exactly one kind of sandwich. Therefore, no more than one kind of sandwich is tasty.

Chapter 13

Natural Deduction Proofs in QL

Trees employ a kind of 'brute force' strategy for proving entailment claims. When the logical structure of the relevant sentences are rather simple, as they are in SL and in some QL cases, it can be an effective strategy, but in some cases they can become very tedious and complex. It is useful to have a proof system that allows one to reason in a more targeted way — especially if you already have an intuitive understanding of why a given argument should be expected to turn out valid. As we did in Chapter 7 for SL, in this chapter we will consider a natural deduction system for quantified logic. As in the case of trees, our QL natural deduction system is an extension of the one we learned previously for SL.

Like our SL natural deduction system, our system can only be used to demonstrate that an argument *is* valid. We do not have a formal method, as part of our natural deduction system, for demonstrating an argument *invalid.* In this respect natural deduction differs from trees. We won't go through the proof in this book, but the natural deduction is both sound and complete. That means that there are natural deductions proofs corresponding to all and only the valid arguments in QL.

13.1 Natural Deduction: the basics

In a natural deduction system, one begins by writing down the assumptions one begins with — these correspond to the premises of an argument — then adds a series of additional sentences, justified via a series of particular rules by the sentences above. Additional assumptions may be made and discharged along the

way. If one succeeds in writing down the conclusion of the argument on a new line, with no additional undischarged assumptions, consistent with the natural deduction rules, one has proven that the conclusion follows from those premises. If you need a refresher on how that works, review Chapter 7.

Our QL system will use all the same sentential rules that our SL system did. All the introduction and elimination rules for the sentential connectives, as well as the derived and replacement rules, will be part of this system too. (These are summarized, along with our new rules, in the Quick Reference guide at the end of this book.) We'll add introduction and elimination rules for the existential and universal quantifiers, and some other derived rules as well.

13.2 Basic quantifier rules

Recall the relationship between quantified sentences and their instances. The sentence Pa is a particular instance of the general claim $\forall x Px$. For any wff Φ, constant c, and variable χ, we define $\Phi\boxed{\chi \Rightarrow c}$ to mean the wff that we get by replacing every occurrence of χ in Φ with c. $\Phi\boxed{\chi \Rightarrow c}$ is called a SUBSTITUTION INSTANCE of $\forall x\Phi$ and $\exists x\Phi$, and c is called the INSTANTIATING CONSTANT. This should be familiar from our discussion of our tree rules for quantifiers in Chapter 10. We will also use this notation to describe our quantifier rules.

13.2.1 Universal elimination

If you have $\forall x Ax$, it is legitimate to infer, of anything, that it is A. You can infer Aa, or Ab, or Az, or Ad_3, etc.. This is, you can infer any substitution instance – in short, you can infer Ac for any constant c. This is the general form of the universal elimination rule ($\forall$E):

$$
\begin{array}{l|l}
m & \forall\chi\Phi \\[4pt]
  & \Phi\boxed{x \Rightarrow c} \qquad \forall\text{E } m
\end{array}
$$

Remember that the box mark for a substitution instance is not a symbol of QL, so you cannot write it directly in a proof. Instead, you write the substituted sentence with the constant c replacing all occurrences of the variable χ in Φ, as in this example:

1	$\forall x(Mx \supset Rxd)$	
2	$Ma \supset Rad$	$\forall$E 1
3	$Md \supset Rdd$	$\forall$E 1

This rule is very similar to the tree rule for universals, which, in our tree system, allowed one to develop a branch containing a universal with any instance of it one likes. Here you are permitted to write down any instance you like on a new line.

13.2.2 Existential introduction

When is it legitimate to infer $\exists xAx$? If you know that something is an A — for instance, if you have Aa available in the proof.

This is the existential introduction rule ($\exists$I):

m	Φ	
	$\exists\chi\Phi \;\boxed{\boxed{c \Rightarrow x}}$	$\exists$I m

It is important to notice that $\Phi\,\boxed{\boxed{c \Rightarrow x}}$ is not necessarily a substitution instance. We write it with double boxes to show that the variable χ does not need to replace all occurrences of the constant c. You can decide which occurrences to replace and which to leave in place. For example, each of lines 2–6 can be justified by $\exists$I:

1	$Ma \supset Rad$	
2	$\exists x(Ma \supset Rax)$	$\exists$I 1
3	$\exists x(Mx \supset Rxd)$	$\exists$I 1
4	$\exists x(Mx \supset Rad)$	$\exists$I 1
5	$\exists y\exists x(Mx \supset Ryd)$	$\exists$I 4
6	$\exists z\exists y\exists x(Mx \supset Ryz)$	$\exists$I 5

13.2.3 Universal introduction

A universal claim like $\forall xPx$ would be proven if every substitution instance of it had been proven, if every sentence Pa, Pb, ... were available in a proof. Alas,

there is no hope of proving *every* substitution instance. That would require proving Pa, Pb, ..., Pj_2, ..., Ps_7, ..., and so on to infinity. There are infinitely many constants in QL, and so this process would never come to an end.

Consider a simple argument: $\forall x Mx$, $\therefore$ $\forall y My$

It makes no difference to the meaning of the sentence whether we use the variable x or the variable y, so this argument is obviously valid. Suppose we begin in this way:

$$
\begin{array}{l|ll}
1 & \forall x Mx & \text{want } \forall y My \\
\hline
2 & Ma & \forall \text{E } 1
\end{array}
$$

We have derived Ma. Nothing stops us from using the same justification to derive Mb, ..., Mj_2, ..., Ms_7, ..., and so on until we run out of space or patience. We have effectively shown the way to prove Mc for any constant c. From this, $\forall y My$ follows.

$$
\begin{array}{l|ll}
1 & \forall x Mx & \\
\hline
2 & Ma & \forall \text{E } 1 \\
3 & \forall y My & \forall \text{I } 2
\end{array}
$$

It is important here that a was just some arbitrary constant. We had not made any special assumptions about it. If a had already been mentioned, say as a premise in the argument, then this would not show anything about *all y*. For example:

$$
\begin{array}{l|ll}
1 & \forall x Rxa & \\
\hline
2 & Raa & \forall \text{E } 1 \\
3 & \forall y Ryy & \text{not allowed!}
\end{array}
$$

This is the schematic form of the universal introduction rule ($\forall$I):

$$
\begin{array}{l|ll}
m & \Phi & \\
& \forall \chi \Phi \boxed{c^* \Rightarrow x} & \forall \text{I } m
\end{array}
$$

* The constant c must not occur in any undischarged assumption.

Note that we can do this for any constant that does not occur in an undischarged assumption and for any variable.

Note also that the constant may not occur in any *undischarged* assumption, but

it may occur as the assumption of a subproof that we have already closed. For example, here is a valid proof of $\forall z(Dz \supset Dz)$ that does not use any premises.

1		Df	want Df
2		Df	R 1
3	$Df \supset Df$		$\supset$I 1–2
4	$\forall z(Dz \supset Dz)$		$\forall$I 3

13.2.4 Existential elimination

A sentence with an existential quantifier tells us that there is *some* member of the UD that satisfies a formula. For example, $\exists x Sx$ tells us (roughly) that there is at least one S. It does not tell us *which* member of the UD satisfies S, however. We cannot immediately conclude Sa, Sf_{23}, or any other substitution instance of the sentence. What can we do?

Suppose that we knew both $\exists x Sx$ and $\forall x(Sx \supset Tx)$. We could reason in this way:

> Since $\exists x Sx$, there is something that is an S. We do not know which constants refer to this thing, if any do, so call this thing 'Ishmael'. From $\forall x(Sx \supset Tx)$, it follows that if Ishmael is an S, then it is a T. Therefore, Ishmael is a T. Because Ishmael is a T, we know that $\exists x Tx$.

In this paragraph, we introduced a name for the thing that is an S. We gave it an arbitrary name ('Ishmael') so that we could reason about it and derive some consequences from there being an S. Since 'Ishmael' is just a bogus name introduced for the purpose of the proof and not a genuine constant, we could not mention it in the conclusion. Yet we could derive a sentence that does not mention Ishmael; namely, $\exists x Tx$. This sentence does follow from the two premises.

We want the existential elimination rule to work in a similar way. Yet since English language worlds like 'Ishmael' are not symbols of QL, we cannot use them in formal proofs. Instead, just as we did in the analogous rule within our tree system, we will use names that are *new* — names which do not appear anywhere else in the proof. (This includes the conclusion you are aiming for.)

A constant that is used to stand in for whatever it is that satisfies an existential claim is called a PROXY. Reasoning with the proxy must all occur inside a

subproof, and the proxy cannot be a constant that is doing work elsewhere in the proof.

This is the schematic form of the existential elimination rule (∃E):

$$
\begin{array}{l|l}
m & \exists \chi \Phi \\[4pt]
n & \quad \boxed{\Phi \;\big|\; x \Rightarrow c^{*}} \qquad\quad \text{for } \exists E \\[4pt]
p & \quad \Psi \\[8pt]
  & \Psi \qquad\qquad\qquad\qquad \exists E \; m,\, n\text{--}p
\end{array}
$$

* The constant c must not appear outside the subproof.

Remember that the proxy constant cannot appear in Ψ, the sentence you prove using ∃E. (It would actually be enough just to require that the proxy constant not appear in $\exists \chi \Phi$, in Ψ, or in any undischarged assumption. In recognition of the fact that it is just a place holder that we use inside the subproof, though, we require an entirely new constant which does not appear anywhere else in the proof.)

The existential elimination rule, like the rules for conditional introduction and negation introduction and elimination, is a rule that involves discharging an assumption. Assume a proxy instance, and see what would follow from that instance; if you have the existential, then, you can stop making the assumption about the proxy, and help yourself to what would have followed from it. As with those other assumption-involving rules, instead of a justification, one includes a note — in this case, 'for ∃E' — about the role of the assumption in the proof. Remember that assumptions must be discharged before your proof is complete, so you should only make an assumption that goes beyond your premises when you have a plan for discharging it.

With this rule, we can give a formal proof that $\exists x S x$ and $\forall x(S x \supset T x)$ together entail $\exists x T x$.

$$
\begin{array}{r|ll}
1 & \exists x S x & \\
2 & \forall x(S x \supset T x) & \text{want } \exists x T x \\
3 & \quad S a & \text{for } \exists E \\
4 & \quad S a \supset T a & \forall E \; 2 \\
5 & \quad T a & \supset E \; 3,\, 4 \\
6 & \quad \exists x T x & \exists I \; 5 \\
7 & \exists x T x & \exists E \; 1,\, 3\text{--}6
\end{array}
$$

13.2.5 Quantifier negation

$\neg\exists\chi\neg\Phi$ is logically equivalent to $\forall\chi\Phi$. The first says that *nothing falsifies Φ*; the second says *everything satisfies Φ*. In QL, they are provably equivalent. Here is a proof schema for half of that equivalence via a natural deduction reductio. For any wff Φ, variable χ, and new name a:

1	$\forall\chi\Phi$	want $\neg\exists\chi\neg\Phi$
2	$\exists\chi\neg\Phi$	for reductio
3	$\neg\Phi\ \boxed{\chi \Rightarrow a^*}$	for $\exists$E
4	$\forall\chi\Phi$	for reductio
5	$\Phi\ \boxed{\chi \Rightarrow a^*}$	$\forall$E 1
6	$\neg\Phi\ \boxed{\chi \Rightarrow a^*}$	R 3
7	$\neg\forall\chi\Phi$	$\neg$I 4–6
8	$\neg\forall\chi\Phi$	$\exists$E 2, 3–7
9	$\forall\chi\Phi$	R 1
10	$\neg\exists\chi\neg\Phi$	$\neg$I 2–9, 2–8

* Where name a does not appear outside the subproof.

This is a proof *schema* — it is not itself a proof in QL, as its lines are not QL sentences. But it describes how a proof of this form can be given. For example, here is one instance of the above schema:

1	$\forall y Ay$	want $\neg\exists y\neg Ay$
2	$\exists y\neg Ay$	for reductio
3	$\neg Ac$	for $\exists$E
4	$\forall y Ay$	for reductio
5	Ac	$\forall$E 1
6	$\neg Ac$	R 3
7	$\neg\forall y Ay$	$\neg$I 4–6
8	$\neg\forall y Ay$	$\exists$E 2, 3–7
9	$\forall y Ay$	R 1
10	$\neg\exists y\neg Ay$	$\neg$I 2–9, 2–8

(Note that this proof encodes the same form of reasoning one would employ to demonstrate via a tree that $\forall y A y \models \neg \exists y \neg A y$. As an exercise: draw out the tree to compare it.)

In order to fully demonstrate that $\neg \exists \chi \neg \Phi$ is logically equivalent to $\forall \chi \Phi$, we would also need a second proof that assumes $\neg \exists \chi \neg \Phi$ and derives $\forall \chi \Phi$. We leave that proof as an exercise for the reader.

It will often be useful to translate between quantifiers by adding or subtracting negations in this way, so we add two derived rules for this purpose. These rules are called quantifier negation (QN):

$$\neg \forall \chi \Phi \iff \exists \chi \neg \Phi$$
$$\neg \exists \chi \Phi \iff \forall \chi \neg \Phi \quad \text{QN}$$

QN is a replacement rule. Like our SL replacement rules (DeMorgan, Double Negation, etc.), it can be used on whole sentences or on subformulae.

13.3 Identity Introduction

The introduction rule for identity is very simple. Everything is identical to itself; so, for any name a, one may write — regardless of what one has on the previous lines of the proof — that $a{=}a$:

$$\left| \; a = a \qquad \text{=I} \right.$$

The =I rule is unlike our other rules in that it does not require referring to any prior lines of the proof. We need only cite the rule itself; it does not reference any previous line numbers.

13.4 Identity Elimination

If you have shown that $a{=}b$, then anything that is true of a must also be true of b. For any sentence with a in it, you can replace some or all of the occurrences of a with b and produce an equivalent sentence. For example, if you already know Raa, then you are justified in concluding Rab, Rba, Rbb. Recall that $\Phi \boxed{\boxed{a \Rightarrow b}}$ is the sentence produced by replacing a in Φ with b. This is not the same as a substitution instance, because b may replace some or all occurrences of a. The identity elimination rule (=E) justifies replacing terms with other terms that are identical to it:

$$
\begin{array}{l|l}
m & c{=}d \\
n & \Phi \\
& \quad \Phi \; \boxed{c \Rightarrow d} \qquad \text{=E } m,\,n \\
& \quad \Phi \; \boxed{d \Rightarrow c} \qquad \text{=E } m,\,n
\end{array}
$$

Here is a simple proof of an instance of the *transitivity of identity*. Let's prove that if $a{=}b$ and $b{=}c$, then $a{=}c$:

$$
\begin{array}{lll}
1 & a{=}b \,\&\, b{=}c & \text{want } a{=}c \\
2 & a{=}b & \&\,\text{E } 1 \\
3 & b{=}c & \&\,\text{E } 1 \\
4 & a{=}c & \text{=E } 2,\,3 \\
5 & (a{=}b \,\&\, b{=}c) \supset a{=}c & \supset\text{I } 1\text{–}4
\end{array}
$$

At line 4, we took advantage of the identity claim $b{=}c$ on line 3, and replaced the b in line 2 with a c. Then we used the familiar $\supset$I rule to discharge the assumption of line 1, proving the conditional we were aiming for.

13.5 Translation and evaluation

Consider this argument: There is only one button in my pocket. There is a blue button in my pocket. So there is no non-blue button in my pocket.

We begin by defining a symbolization key:

> UD: buttons in my pocket
> Bx: x is blue.

Because we have no need to discuss anything other than buttons in my pocket, we've restricted the UD accordingly. If we included other things (buttons elsewhere and/or things other than buttons), we'd need predicates corresponding to being a button and things' locations. The simple version here is adequate for our present needs. The argument is translated as:

1. $\forall x \forall y \; x{=}y$
2. $\exists x Bx$
3. $\therefore \; \neg \exists x \neg Bx$

So the set-up for a natural deduction proof will be:

1 | $\forall x \forall y \; x{=}y$

2 | $\exists x Bx$ want $\neg\exists x\neg Bx$

3 |

There are various strategies one might employ. Here are two clues that point toward one promising strategy. Note again that we have an existential on line 2 — this suggests existential elimination as a possible strategy. Note also that we are aiming for $\neg\exists x\neg Bx$, which equivalent to $\neg\neg\forall x Bx$ by QN. This in turn is equivalent, by DN, to $\forall x Bx$, which suggests that universal introduction is going to be an important step. If we introduce an assumption with a proxy instance of $\exists x Bx$, we'll be able to work toward a generic instance of Bx. In this example, we'll take e as our proxy, and show that Bf follows from $\exists x Bx$:

1 | $\forall x \forall y \; x{=}y$

2 | $\exists x Bx$ want $\neg\exists x\neg Bx$

3 | Be for $\exists$E

4 | $\forall y \; e{=}y$ $\forall$E 1

5 | $e{=}f$ $\forall$E 4

6 | Bf =E 5, 3

7 | Bf $\exists$E 2, 3–6

By line 7, we have discharged the assumption about the proxy — we won't use the name e any more — we have established that Bf follows from the two premises. Since f is an arbitrary name — one that does not appear in any undischarged assumption — we can perform universal introduction on that instance. This in turn lets us complete the proof via the two substitution rules mentioned above:

1	$\forall x \forall y\ x{=}y$	
2	$\exists x Bx$	want $\neg\exists x\neg Bx$
3	$\quad Be$	for $\exists$E
4	$\quad \forall y\ e{=}y$	$\forall$E 1
5	$\quad e{=}f$	$\forall$E 4
6	$\quad Bf$	=E 5, 3
7	Bf	$\exists$E 2, 3–6
8	$\forall x Bx$	$\forall$I 7
9	$\neg\neg\forall x Bx$	DN 8
10	$\neg\exists x\neg Bx$	QN 9

13.6 Natural deduction strategy

All the strategy advice given in §7.12 is equally applicable to natural deduction proofs in QL. Review the suggestions there for general advice for natural deduction proofs. Applied to our new QL rules, if you have a universal, you can think about taking any instances that look useful. (Taking random instances is unlikely to be useful.) If you have an existential, consider using the existential elimination rule, which begins by assuming an instance with a proxy, then deriving a conclusion that does not contain that proxy name.

Showing $\neg\exists x\Phi$ can also be hard, and it is often easier to show $\forall x\neg\Phi$ and use the QN rule.

13.7 Soundness and completeness

The proofs for soundness and completeness of our natural deduction system are beyond the scope of this textbook. But if you are interested in thinking through how those proofs would go, here are a few hints to get you started. Soundness in a natural deduction system amounts to the claim that if any sentence Φ is derivable from a set of sentences $\mathcal{X}$, then $\mathcal{X} \models \Phi$. To prove this, you would need to demonstrate that any possible natural deduction proof meets this constraint. This is trivial for 'proofs' that only contain premises; you'd next have to show, for every possible way of extending the proof (i.e., everything permitted by any one of our rules), that any newly added lines with no undischarged assumptions

are entailed by the premises.

Undischarged assumptions would require special treatment. You can think of an assumption as being similar to a 'temporary premise' — what you really want to prove is that, for every possible line in a proof, that line is entailed by the premises *in addition to* any undischarged assumptions.

The completeness proof is more complex. We need some way to guarantee that there is a proof corresponding to every QL entailment. The way to do this is to find an algorithmic procedure that is guaranteed to find a proof if one exists, and to prove that this is so. One good way to do this is to take advantage of the proven completeness of our *tree* system for QL, presented in Chapter 11, and find a way to demonstrate that any tree proof can be converted to a natural deduction proof. Here is a hint if you'd like to undertake that project: the tree method encodes the same kind of reasoning that reductio proofs do.

Practice Exercises

$\star$ **Part A** Provide a justification (rule and line numbers) for each line of proof that requires one.

1	$\forall x \exists y (Rxy \lor Ryx)$
2	$\forall x \neg Rmx$
3	$\exists y (Rmy \lor Rym)$
4	$\quad Rma \lor Ram$
5	$\quad \neg Rma$
6	$\quad Ram$
7	$\quad \exists x Rxm$
8	$\exists x Rxm$

1	$\forall x (\exists y Lxy \supset \forall z Lzx)$
2	Lab
3	$\exists y Lay \supset \forall z Lza$
4	$\exists y Lay$
5	$\forall z Lza$
6	Lca
7	$\exists y Lcy \supset \forall z Lzc$
8	$\exists y Lcy$
9	$\forall z Lzc$
10	Lcc
11	$\forall x Lxx$

1	$\forall x(Jx \supset Kx)$		1	$\neg(\exists x Mx \vee \forall x \neg Mx)$
2	$\exists x \forall y Lxy$		2	$\neg \exists x Mx \,\&\, \neg \forall x \neg Mx$
3	$\forall x Jx$		3	$\neg \exists x Mx$
4	$\forall y Lay$		4	$\forall x \neg Mx$
5	Ja		5	$\neg \forall x \neg Mx$
6	$Ja \supset Ka$		6	$\exists x Mx \vee \forall x \neg Mx$
7	Ka			
8	Laa			
9	$Ka \,\&\, Laa$			
10	$\exists x(Kx \,\&\, Lxx)$			
11	$\exists x(Kx \,\&\, Lxx)$			

⋆ **Part B** Provide a natural deduction proof of each claim.

1. $\vdash \forall x Fx \vee \neg \forall x Fx$
2. $\{\forall x(Mx \equiv Nx), Ma \,\&\, \exists x Rxa\} \vdash \exists x Nx$
3. $\{\forall x(\neg Mx \vee Ljx), \forall x(Bx \supset Ljx), \forall x(Mx \vee Bx)\} \vdash \forall x Ljx$
4. $\forall x(Cx \,\&\, Dt) \vdash \forall x Cx \,\&\, Dt$
5. $\exists x(Cx \vee Dt) \vdash \exists x Cx \vee Dt$

Part C Provide a proof of the argument about Billy on p. .

Part D Look back at Part . Provide proofs to show that each of the argument forms is valid in QL.

⋆ **Part E** Provide a natural deduction proof of each claim.

1. $\forall x \forall y Gxy \vdash \exists x Gxx$
2. $\forall x \forall y(Gxy \supset Gyx) \vdash \forall x \forall y(Gxy \equiv Gyx)$
3. $\{\forall x(Ax \supset Bx), \exists x Ax\} \vdash \exists x Bx$
4. $\{Na \supset \forall x(Mx \equiv Ma), Ma, \neg Mb\} \vdash \neg Na$
5. $\vdash \forall z(Pz \vee \neg Pz)$
6. $\vdash \forall x Rxx \supset \exists x \exists y Rxy$
7. $\vdash \forall y \exists x(Qy \supset Qx)$

Part F Show that each pair of sentences is provably equivalent.

1. $\forall x(Ax \supset \neg Bx)$, $\neg\exists x(Ax \,\&\, Bx)$
2. $\forall x(\neg Ax \supset Bd)$, $\forall x Ax \lor Bd$
3. $\exists x Px \supset Qc$, $\forall x(Px \supset Qc)$

Part G Show that each of the following is provably inconsistent.

1. $\{Sa \supset Tm,\ Tm \supset Sa,\ Tm \,\&\, \neg Sa\}$
2. $\{\neg\exists x Rxa,\ \forall x\forall y Ryx\}$
3. $\{\neg\exists x\exists y Lxy,\ Laa\}$
4. $\{\forall x(Px \supset Qx),\ \forall z(Pz \supset Rz),\ \forall y Py,\ \neg Qa \,\&\, \neg Rb\}$

⋆ **Part H** Write a symbolization key for the following argument, translate it, and prove it:

> There is someone who likes everyone who likes everyone that first person likes. Therefore, there is someone who likes themself.

Part I Provide a proof of each claim.

1. $\{Pa \lor Qb, Qb \supset b{=}c, \neg Pa\} \vdash Qc$
2. $\{m{=}n \lor n{=}o, An\} \vdash Am \lor Ao$
3. $\{\forall x\ x{=}m, Rma\} \vdash \exists x Rxx$
4. $\neg\exists x\ x{\neq}m \vdash \forall x\forall y(Px \supset Py)$
5. $\forall x\forall y(Rxy \supset x{=}y) \vdash Rab \supset Rba$
6. $\{\exists x Jx, \exists x\neg Jx\} \vdash \exists x\exists y\ x{\neq}y$
7. $\{\forall x(x{=}n \equiv Mx), \forall x(Ox \lor \neg Mx)\} \vdash On$
8. $\{\exists x Dx, \forall x(x{=}p \equiv Dx)\} \vdash Dp$
9. $\{\exists x\big[Kx \,\&\, \forall y(Ky \supset x{=}y) \,\&\, Bx\big], Kd\} \vdash Bd$
10. $\vdash Pa \supset \forall x(Px \lor x{\neq}a)$

⋆ **Part J** For each of the following pairs of sentences: If they are logically equivalent in QL, give proofs to show this. If they are not, construct a model to show this.

1. $\forall x Px \supset Qc$, $\forall x(Px \supset Qc)$
2. $\forall x Px \,\&\, Qc$, $\forall x(Px \,\&\, Qc)$
3. $Qc \lor \exists x Qx$, $\exists x(Qc \lor Qx)$
4. $\forall x\forall y\forall z Bxyz$, $\forall x Bxxx$
5. $\forall x\forall y Dxy$, $\forall y\forall x Dxy$
6. $\exists x\forall y Dxy$, $\forall y\exists x Dxy$

⋆ **Part K** For each of the following arguments: If it is valid in QL, give a proof. If it is invalid, construct a model to show that it is invalid.

1. $\forall x \exists y Rxy$, $\therefore$ $\exists y \forall x Rxy$
2. $\exists y \forall x Rxy$, $\therefore$ $\forall x \exists y Rxy$
3. $\exists x (Px \,\&\, \neg Qx)$, $\therefore$ $\forall x (Px \supset \neg Qx)$
4. $\forall x (Sx \supset Ta)$, Sd, $\therefore$ Ta
5. $\forall x (Ax \supset Bx)$, $\forall x (Bx \supset Cx)$, $\therefore$ $\forall x (Ax \supset Cx)$
6. $\exists x (Dx \lor Ex)$, $\forall x (Dx \supset Fx)$, $\therefore$ $\exists x (Dx \,\&\, Fx)$
7. $\forall x \forall y (Rxy \lor Ryx)$, $\therefore$ Rjj
8. $\exists x \exists y (Rxy \lor Ryx)$, $\therefore$ Rjj
9. $\forall x Px \supset \forall x Qx$, $\exists x \neg Px$, $\therefore$ $\exists x \neg Qx$
10. $\exists x Mx \supset \exists x Nx$, $\neg \exists x Nx$, $\therefore$ $\forall x \neg Mx$

Part L Look at the arguments given in Chapter 10, Problem Part C (page 200). For those arguments whose QL translations are valid, prove their validity via natural deduction.

Part M Look at the entailment claims given in Chapter 12, Problem Part F (page 227). For those entailment claims that are true, prove them via natural deduction.

Part N Look at the arguments given in Chapter 12, Problem Part H (page 227). For those arguments whose QL translations are valid, prove their validity via natural deduction.

Appendix A

Symbolic notation

In the history of formal logic, different symbols have been used at different times and by different authors. Often, authors were forced to use notation that their printers could typeset.

In one sense, the choice of symbols used for various logical constants is arbitrary. There is nothing written in heaven that says that '¬' must be the symbol for truth-functional negation. We might have specified a different symbol to play that part. Once we have given definitions for well-formed formulae (wff) and for truth in our logic languages, however, using '¬' is no longer arbitrary. That is the symbol for negation in this textbook, and so it is the symbol for negation when writing sentences in our languages SL or QL.

This appendix presents some common symbols, so that you can recognize them if you encounter them in an article or in another book.

summary of symbols	
negation	¬, ∼
conjunction	&, ∧, •
disjunction	∨
conditional	→, ⊃
biconditional	↔, ≡

Negation   Two commonly used symbols are the *hoe*, '¬', and the *swung dash*, '∼.' In some more advanced formal systems it is necessary to distinguish between two kinds of negation; the distinction is sometimes represented by using both '¬' and '∼.'

Disjunction   The symbol '∨' is typically used to symbolize inclusive disjunction.

Conjunction   Conjunction is often symbolized with the *ampersand*, '&.' The ampersand is actually a decorative form of the Latin word 'et' which means 'and'; it is commonly used in English writing. As a symbol in a formal system, the ampersand is not the word 'and'; its meaning is given by the formal semantics for the language. Perhaps to avoid this confusion, some systems use a different symbol for conjunction. For example, '∧' is a counterpart to the symbol used for disjunction. Sometimes a single dot, '•', is used. In some older texts, there is no symbol for conjunction at all; 'A and B' is simply written 'AB.'

Material Conditional   There are two common symbols for the material conditional: the *arrow*, '→', and the *hook*, '⊃.'

Material Biconditional   The *double-headed arrow*, '↔', is used in systems that use the arrow to represent the material conditional. Systems that use the hook for the conditional typically use the *triple bar*, '≡', for the biconditional.

Quantifiers   The universal quantifier is typically symbolized as an upside-down A, '∀', and the existential quantifier as a backwards E, '∃.' Quantifiers are given in some systems in parentheses, as in '$(\forall x)(\exists x)Fxy$', and in some systems without, as in '$\forall x \exists x Fxy$'.

In some texts, there is no separate symbol for the universal quantifier. Instead, the variable is just written in parentheses in front of the formula that it binds. For example, 'all x are P' is written $(x)Px$.

In some systems, the quantifiers are symbolized with larger versions of the symbols used for conjunction and disjunction. Although quantified expressions cannot be translated into expressions without quantifiers, there is a conceptual connection between the universal quantifier and conjunction and between the existential quantifier and disjunction. Consider the sentence $\exists x Px$, for example. It means that *either* the first member of the UD is a P, *or* the second one is, *or* the third one is, ... Such a system uses the symbol '⋁' instead of '∃.'

Polish notation

This section briefly discusses sentential logic in Polish notation, a system of notation introduced in the late 1920s by the Polish logician Jan Łukasiewicz.

notation of SL	Polish notation
¬	N
&	K
∨	A
⊃	C
≡	E

Lower case letters are used as sentence letters. The capital letter N is used for negation. A is used for disjunction, K for conjunction, C for the conditional, E for the biconditional. ('A' is for alternation, another name for logical disjunction. 'E' is for equivalence.)

In Polish notation, a binary connective is written *before* the two sentences that it connects. For example, the sentence $A \,\&\, B$ of SL would be written Kab in Polish notation.

The sentences $\neg A \supset B$ and $\neg(A \supset B)$ are very different; the main logical operator of the first is the conditional, but the main connective of the second is negation. In SL, we show this by putting parentheses around the conditional in the second sentence. In Polish notation, parentheses are never required. The left-most connective is always the main connective. The first sentence would simply be written $CNab$ and the second $NCab$.

This feature of Polish notation means that it is possible to evaluate sentences simply by working through the symbols from right to left. If you were constructing

a truth table for $NKab$, for example, you would first consider the truth-values assigned to b and a, then consider their conjunction, and then negate the result. The general rule for what to evaluate next in SL is not nearly so simple. In SL, the truth table for $\neg(A \mathbin{\&} B)$ requires looking at A and B, then looking in the middle of the sentence at the conjunction, and then at the beginning of the sentence at the negation. Because the order of operations can be specified more mechanically in Polish notation, variants of Polish notation are used as the internal structure for many computer programming languages.

Appendix B

Solutions to selected exercises

Many of the exercises may be answered correctly in different ways. Where that is the case, the solution here represents one possible correct answer.

Chapter 1

Chapter 1 Part A

1, 2, 4, 5, 6, 8, and 10 are sentences.

Chapter 1 Part C

1. consistent
2. inconsistent
3. consistent
4. consistent

Chapter 1 Part D

1. Possible. For example:

    Vancouver is in Canada.
    Seattle is in Canada.
    ∴ Seattle and Vancouver are both in Canada.

2. Possible. The previous argument is an example.
3. Possible. For example:

    Vancouver is in Canada.
    Vancouver is not in Canada
    ∴ Vancouver is in Canada and it is not in Canada.

4. Impossible. If the conclusion is a tautology, the argument is guaranteed to be valid.
5. Impossible. By definition, a tautology is not contingent.
6. Possible. Any two tautologies will do.
7. Impossible. Anything logically equivalent to a tautology is a tautology.
8. Possible. For example:

> Vancouver is in Canada and Vancouver is not in Canada.
> Seattle is in Canada and Seattle is not in Canada.

9. Impossible. If it contains a contradiction, it is inconsistent.
10. Possible. For example:

> Vancouver is in Canada or Vancouver is not in Canada.
> Seattle is in Canada and Seattle is not in Canada.

Chapter 2

Chapter 2 Part A

1. $\neg M$
2. $M \vee \neg M$
3. $G \vee C$
4. $\neg C \mathbin{\&} \neg G$
5. $C \supset (\neg G \mathbin{\&} \neg M)$
6. $M \vee (C \vee G)$

Chapter 2 Part C

1. $E_1 \mathbin{\&} E_2$
2. $F_1 \supset S_1$
3. $F_1 \vee E_1$
4. $E_2 \mathbin{\&} \neg S_2$
5. $\neg E_1 \mathbin{\&} \neg E_2$
6. $E_1 \mathbin{\&} E_2 \mathbin{\&} \neg(S_1 \vee S_2)$
7. $S_2 \supset F_2$
8. $(\neg E_1 \supset \neg E_2) \mathbin{\&} (E_1 \supset E_2)$
9. $S_1 \equiv \neg S_2$
10. $(E_2 \mathbin{\&} F_2) \supset S_2$
11. $\neg(E_2 \mathbin{\&} F_2)$
12. $(F_1 \mathbin{\&} F_2) \equiv (\neg E_1 \mathbin{\&} \neg E_2)$

Chapter 2 Part D

A: Alice is a spy.
B: Bob is a spy.
C: The code has been broken.

G: The German embassy will be in an uproar.

1. $A \& B$
2. $(A \vee B) \supset C$
3. $\neg(A \vee B) \supset \neg C$
4. $G \vee C$
5. $(C \vee \neg C) \& G$
6. $(A \vee B) \& \neg(A \& B)$

Chapter 2 Part E

G: Gregor plays first base.
L: The team will lose.
M: There is a miracle.
C: Gregor's mom will bake cookies.

1. $G \supset L$
2. $\neg M \supset L$
3. $(L \vee \neg L) \& G$
4. $C \equiv G$
5. $M \supset \neg C$

Chapter 2 Part G

1. (a) no (b) no
2. (a) no (b) yes
3. (a) yes (b) yes
4. (a) no (b) no
5. (a) yes (b) yes
6. (a) no (b) no
7. (a) no (b) yes
8. (a) no (b) yes
9. (a) no (b) no

Chapter 3

Chapter 3 Part A

1. tautology
2. contradiction

3. contingent
4. tautology
5. tautology
6. contingent
7. tautology
8. contradiction
9. tautology
10. contradiction
11. tautology
12. contingent
13. contradiction
14. contingent
15. tautology
16. tautology
17. contingent
18. contingent

Chapter 3 Part B

2, 3, 5, 6, 8, and 9 are logically equivalent.

Chapter 3 Part C

2, 4, 7, 8, and 9 are consistent.

Chapter 3 Part D

3, 5, 8, and 10 are valid.

Chapter 3 Part E

1. Φ and Ψ have the same truth value on every line of a complete truth table, so $\Phi \equiv \Psi$ is true on every line. It is a tautology.
2. The sentence is false on some line of a complete truth table. On that line, Φ and Ψ are true and Ω is false. So the argument is invalid.
3. Since there is no line of a complete truth table on which all three sentences are true, the conjunction is false on every line. So it is a contradiction.
4. Since Φ is false on every line of a complete truth table, there is no line on which Φ and Ψ are true and Ω is false. So the argument is valid.
5. Since Ω is true on every line of a complete truth table, there is no line on which Φ and Ψ are true and Ω is false. So the argument is valid.

6. Not much. $(\varPhi \vee \varPsi)$ is a tautology if $\varPhi$ and $\varPsi$ are tautologies; it is a contradiction if they are contradictions; it is contingent if they are contingent.
7. $\varPhi$ and $\varPsi$ have different truth values on at least one line of a complete truth table, and $(\varPhi \vee \varPsi)$ will be true on that line. On other lines, it might be true or false. So $(\varPhi \vee \varPsi)$ is either a tautology or it is contingent; it is *not* a contradiction.

Chapter 3 Part F

1. $\neg A \supset B$
2. $\neg(A \supset \neg B)$
3. $\neg[(A \supset B) \supset \neg(B \supset A)]$

Chapter 5

Chapter 5 Part A

1. (a) $\{P, P \supset Q, Q \supset \neg P\}$, (b) true
2. (a) $\neg((P \supset Q) \equiv (Q \supset P))$, (b) true
3. (a) $\{P \,\&\, Q, \neg R \supset \neg Q, \neg(P \,\&\, R)\}$, (b) true
4. (a) $\{A \vee B, B \supset C, A \equiv C, \neg C\}$, (b) true
5. (a) $A \equiv \neg A$, (b) true
6. (a) $\{P, P \supset Q, \neg Q, \neg A\}$, (b) true
7. (a) $\{P \supset Q, \neg P \vee \neg Q, Q \supset P\}$, (b) false.

Chapter 5 Part B

The argument is valid.

$\{A \equiv B, \neg B \supset (C \vee D), E \supset \neg C, (\neg D \,\&\, F) \vee G, \neg A \,\&\, E\} \vdash H \vee G$

1.	$A \equiv B$ ✓	
2.	$\neg B \supset (C \vee D)$ ✓	
3.	$E \supset \neg C$ ✓	
4.	$(\neg D \,\&\, F) \vee G$ ✓	
5.	$\neg A \,\&\, E$ ✓	
6.	$\neg(H \vee G)$ ✓	

7.	$\neg A$	5 &
8.	E	

9.	$\neg H$	6 ¬∨
10.	$\neg G$	

11.	$\neg D \,\&\, F$ ✓ G		4 ∨
	$\times$		
12.	$\neg D$		11 &
13.	F		
14.	$\neg E$ $\neg C$		3 ⊃
	$\times$		
15.	A $\neg A$		1 ≡
16.	B $\neg B$		
	$\times$		
17.	$\neg\neg B$ $C \vee D$ ✓		2 ⊃
	$\times$		
18.	C D		17 ∨
	$\times$ $\times$		

Chapter 5 Part C

1. True.
   $\{P, P \supset Q, Q \supset \neg P\} \vdash \bot$

1.	P
2.	$P \supset Q$ ✓
3.	$Q \supset \neg P$ ✓

   4. $\neg P$ Q
          ×

   5. $\neg Q$ $\neg P$
            ×   ×

2. False.

   1. $\neg((P \supset Q) \equiv (Q \supset P))$ ✓

   2. $P \supset Q$ ✓ $\neg(P \supset Q)$ ✓   1 ¬≡
   3. $\neg(Q \supset P)$ ✓ $Q \supset P$ ✓

   4. Q 3 ¬⊃
   5. $\neg P$

   6. P 2 ¬⊃
   7. $\neg P$ Q $\neg Q$ 2 ⊃
        ↑   ↑

   8. $\neg Q$ P 3 ⊃
              ↑   ↑
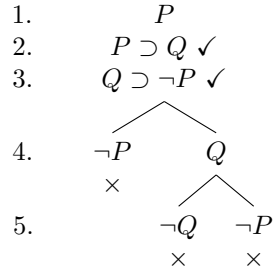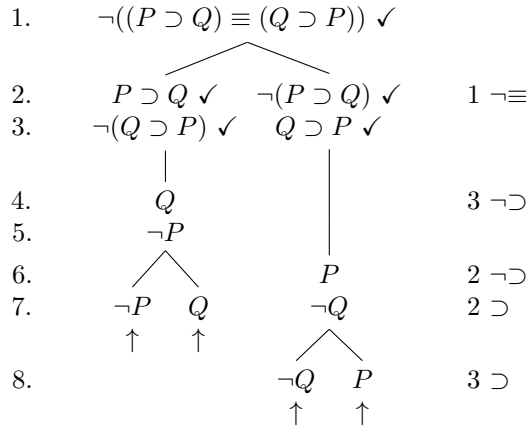
   $\{P = 0, Q = 1\}$ and $\{P = 1, Q = 0\}$ each falsify $((P \supset Q) \equiv (Q \supset P))$, so it is not a tautology.

3. True.

1.	$P \,\&\, Q$ ✓
2.	$\neg R \supset \neg Q$ ✓
3.	$\neg(P \,\&\, R)$ ✓

   4. P 1 &
   5. Q

   6. $\neg P$ $\neg R$ 3 ¬&
        ×

   7. $\neg\neg R$ $\neg Q$ 2 ⊃
             ×   ×

4. True.

```
1.                    A ∨ B ✓
2.                    B ⊃ C ✓
3.                    A ≡ C ✓
4.                      ¬C

5.              ¬B          C        2 ⊃
                           ×
6.         A       B                 1 ∨
                   ×
7.      A    ¬A                      3 ≡
8.      C    ¬C
        ×    ×
```

5. True.

```
1.      A ≡ ¬A ✓

2.       A    ¬A        1 ≡
3.      ¬A    A
        ×     ×
```

6. True.

```
1.       P ✓
2.      P ⊃ Q ✓
3.       ¬Q ✓
4.        ¬A

5.     ¬P    Q        2 ⊃
        ×    ×
```
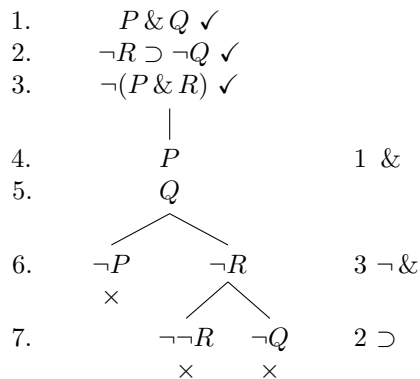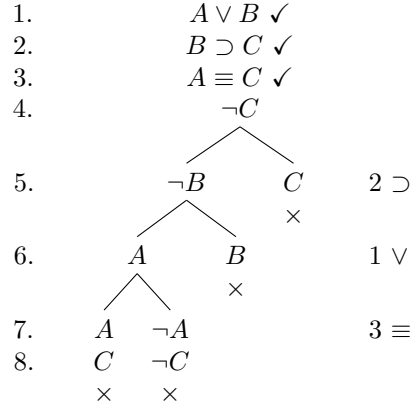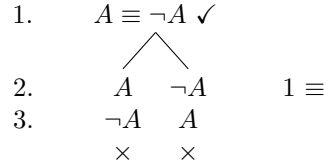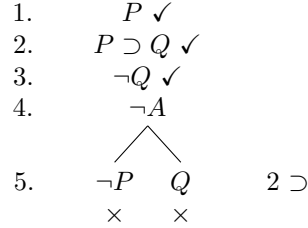
7. True. $\{P = 0, Q = 0\}$ satisfies these sentences.

```
1.                          P ⊃ Q ✓
2.                         ¬P ∨ ¬Q ✓
3.                          Q ⊃ P ✓

4.              ¬P                  ¬Q        2 ∨

5.         ¬P         Q         ¬P      Q     1 ⊃
                                        ×
6.     ¬Q    P    ¬Q    P    ¬Q    P          3 ⊃
        ↑    ×    ×     ×     ↑    ×
```
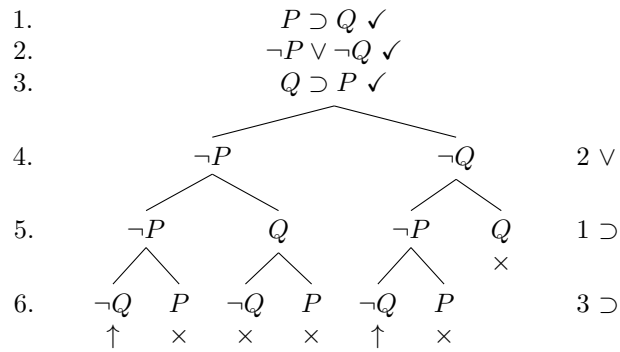
Chapter 6

Chapter 6 Part A

1. Yes. Assuming that the root $\Phi \,\&\, \Psi$ is satisfiable, for some interpretation $\mathcal{I}$, $\mathcal{I}(\Phi \,\&\, \Psi) = 1$, so $\mathcal{I}(\Phi) = 1$ and $\mathcal{I}(\Psi) = 1$. So if what comes above is satisfiable, at least one branch is guaranteed to be satisfiable after this rule has been performed. (In fact, both are.)

2. Yes. Assuming that the root $\Phi \,\&\, \Psi$ is satisfiable, for some interpretation $\mathcal{I}$, $\mathcal{I}(\Phi \,\&\, \Psi) = 1$, so $\mathcal{I}(\Phi) = 1$. So if what comes above is satisfiable, the continuation is guaranteed to be satisfiable after this rule has been performed.

3. No. It is possible to satisfy the root $\Phi \,\&\, \Psi$ without satisfying the extension of the tree. Here is a counterexample to the soundness of the revised system. This tree has a satisfiable root, but closes if we use the rule in question.

$$
\begin{array}{c}
B \\
A \,\&\, B \; \checkmark \\
| \\
A \\
\neg B \\
\times
\end{array}
$$

4. No. It is possible to satisfy the root $\Phi \lor \Psi$ without satisfying the extension of the tree. Here is a counterexample to the soundness of the revised system. This tree has a satisfiable root, but closes if we use the rule in question.

$$
\begin{array}{c}
\neg A \\
A \lor B \; \checkmark \\
| \\
A \\
B \\
\times
\end{array}
$$

5. Yes. Assume $\mathcal{I}(\Phi \lor \Psi) = 1$. Then either $\mathcal{I}(\Phi) = 1$ or $\mathcal{I}(\Psi) = 1$. If $\mathcal{I}(\Phi) = 1$, then $\mathcal{I}$ satisfies the left branch. If $\mathcal{I}(\Psi) = 1$, then $\mathcal{I}$ satisfies the middle branch. So, assuming that $\mathcal{I}$ satisfies the sentences above this resolution rule, it must satisfy at least one branch below it. So this rule will never take us from a satisfiable branch to a tree with no satisfiable branches.

6. Yes. Assume $\mathcal{I}(\Phi \supset \Psi) = 1$. Then either $\mathcal{I}(\neg\Phi) = 1$ or $\mathcal{I}(\Psi) = 1$. If $\mathcal{I}(\neg\Phi) = 1$, then $\mathcal{I}(\Phi) = 0$, so $\mathcal{I}$ satisfies the left branch. If $\mathcal{I}(\Psi) = 1$, then $\mathcal{I}$ satisfies the right branch, as the rule for disjunctions is if at least one disjunct is satisfied, the disjunction is satisfied. So, assuming that $\mathcal{I}$ satisfies the sentences above this resolution rule, it must satisfy at least one branch below it. So this rule will never take us from a satisfiable branch to a tree with no satisfiable branches.

7. Yes. Assume $\mathcal{I}(\Phi \supset \Psi) = 1$. Then either $\mathcal{I}(\neg\Phi) = 1$ or $\mathcal{I}(\Psi) = 1$. In either case, the single branch is satisfied, as the rule for disjunctions is if at least one disjunct is satisfied, the disjunction is satisfied. So, assuming that $\mathcal{I}$ satisfies the sentences above this resolution rule, it must satisfy at least one branch below it. So this rule will never take us from a satisfiable branch to a tree with no satisfiable branches.

8. No. Here is a counterexample to the soundness of the revised system. This tree has a satisfiable root, but closes if we use the rule in question.

$$P \equiv Q \checkmark$$
$$\neg P$$
$$\vert$$
$$P$$
$$Q$$
$$\times$$

9. Yes. Assume $\mathcal{I}(\Phi \vee \Psi) = 1$. Then either $\mathcal{I}(\Phi) = 1$ or $\mathcal{I}(\Psi) = 1$. If $\mathcal{I}(\Phi) = 1$, then $\mathcal{I}$ satisfies the left branch. If $\mathcal{I}(\Psi) = 1$, then $\mathcal{I}$ satisfies the middle branch. So, assuming that $\mathcal{I}$ satisfies the sentences above this resolution rule, it must satisfy at least one branch below it. So this rule will never take us from a satisfiable branch to a tree with no satisfiable branches.

Chapter 7

Chapter 7 Part A

1	$W \supset \neg B$	
2	$A \& W$	
3	$B \vee (J \& K)$	
4	W	& E 2
5	$\neg B$	$\supset$E 1, 4
6	$J \& K$	$\vee$E 3, 5
7	K	& E 6

1	$L \equiv \neg O$		
2	$L \vee \neg O$		
3		$\neg L$	for reductio
4		$\neg O$	$\vee$E 2, 3
5		L	$\equiv$E 1, 4
6		$\neg L$	R 3
7	L		$\neg$E 3–5, 3–6

1	$Z \supset (C \,\&\, \neg N)$	
2	$\neg Z \supset (N \,\&\, \neg C)$	
3	$\neg(N \lor C)$	for reductio
4	$\neg N \,\&\, \neg C$	DeM 3
5	Z	for reductio
6	$C \,\&\, \neg N$	$\supset$E 1, 5
7	C	& E 6
8	$\neg C$	& E 4
9	$\neg Z$	$\neg$I 5–7, 5–8
10	$N \,\&\, \neg C$	$\supset$E 2, 9
11	N	& E 10
12	$\neg N$	& E 4
13	$N \lor C$	$\neg$E 3–11, 3–12

Chapter 7 Part B

1.

1	$K \,\&\, L$	want $K \equiv L$
2	K	want L
3	L	& E 1
4	$K \supset L$	$\supset$I 2–3
5	L	want K
6	K	& E 1
7	$L \supset K$	$\supset$I 5–6
8	$K \equiv L$	$\equiv$I 4, 7

2.

	1	$A \supset (B \supset C)$	want $(A \& B) \supset C$
	2	$A \& B$	want C
	3	A	$\& E$ 2
	4	$B \supset C$	$\supset E$ 1, 3
	5	B	$\& E$ 2
	6	C	$\supset E$ 4, 5
	7	$(A \& B) \supset C$	$\supset I$ 2–6

3.

	1	$P \& (Q \vee R)$	
	2	$P \supset \neg R$	want $Q \vee E$
	3	P	$\& E$ 1
	4	$\neg R$	$\supset E$ 2, 3
	5	$Q \vee R$	$\& E$ 1
	6	Q	$\vee E$ 4, 5
	7	$Q \vee E$	$\vee I$ 6

4.

	1	$(C \& D) \vee E$	want $E \vee D$
	2	$\neg E$	want D
	3	$C \& D$	$\vee E$ 1, 2
	4	D	$\& E$ 3
	5	$\neg E \supset D$	$\supset I$ 2–4
	6	$E \vee D$	MC 5

5.

1	$\neg F \supset G$	
2	$F \supset H$	want $G \vee H$
3	$\neg G$	want H
4	$\neg\neg F$	MT 1, 3
5	F	DN 4
6	H	$\supset$E 2, 5
7	$\neg G \supset H$	$\supset$I 3–6
8	$G \vee H$	MC 7

6.

1	$(X \,\&\, Y) \vee (X \,\&\, Z)$	
2	$\neg(X \,\&\, D)$	
3	$D \vee M$	want M
4	$\neg X$	for reductio
5	$\neg X \vee \neg Y$	$\vee$I 4
6	$\neg(X \,\&\, Y)$	DeM 5
7	$X \,\&\, Z$	$\vee$E 1, 6
8	X	$\&$E 7
9	$\neg X$	R 4
10	X	$\neg$E 4–8, 4–9
11	$\neg M$	for reductio
12	D	$\vee$E 3, 11
13	$X \,\&\, D$	$\&$I 10, 12
14	$\neg(X \,\&\, D)$	R 2
15	M	$\neg$E 11–13, 11–14

Chapter 7 Part D

1.
$$
\begin{array}{ll}
& 1 \quad\ \ \ \left|\ \ \right| O \qquad\qquad \text{want } O \\
O \supset O\ \ & 2 \quad\ \ \ \left|\ \ \right| O \qquad\qquad \text{R } 1 \\
& 3 \quad\ \ \ \left| O \supset O \qquad\quad \supset\!\text{I } 1\text{--}2 \right.
\end{array}
$$

2.
$$
\begin{array}{ll}
& 1 \quad\ \ \left|\ \ \right| \neg(N \vee \neg N) \qquad \text{for reductio} \\
& 2 \quad\ \ \left|\ \ \right| \neg N \,\&\, \neg\neg N \qquad \text{DeM } 1 \\
N \vee \neg N\ \ & 3 \quad\ \ \left|\ \ \right| \neg N \qquad\qquad\quad \&\,\text{E } 2 \\
& 4 \quad\ \ \left|\ \ \right| \neg\neg N \qquad\qquad \&\,\text{E } 2 \\
& 5 \quad\ \ \left| N \vee \neg N \qquad\qquad\ \ \neg\text{E } 1\text{--}3,\ 1\text{--}4 \right.
\end{array}
$$

3.
$$
\begin{array}{ll}
& 1 \quad\ \ \left|\ \ \right| P \,\&\, \neg P \qquad \text{for reductio} \\
& 2 \quad\ \ \left|\ \ \right| P \qquad\qquad\ \&\,\text{E } 1 \\
\neg(P \,\&\, \neg P)\ \ & 3 \quad\ \ \left|\ \ \right| \neg P \qquad\qquad \&\,\text{E } 1 \\
& 4 \quad\ \ \left| \neg(P \,\&\, \neg P) \qquad \neg\text{I } 1\text{--}2,\ 1\text{--}3 \right.
\end{array}
$$

4.
$$
\begin{array}{ll}
1 \quad\left|\ \ \right| \neg(A \supset \neg C) & \text{want } A \supset C \\
2 \quad\left|\ \ \right|\ \left| A \right. & \text{want } C \\
3 \quad\left|\ \ \right|\ \left| \neg(\neg A \vee \neg C) \right. & \text{MC } 1 \\
4 \quad\left|\ \ \right|\ \left| \neg\neg A \,\&\, \neg\neg C \right. & \text{DeM } 3 \\
5 \quad\left|\ \ \right|\ \left| \neg\neg A \,\&\, C \right. & \text{DN } 4 \\
6 \quad\left|\ \ \right|\ \left| C \right. & \&\,\text{E } 5 \\
7 \quad\left|\ \ \right| A \supset C & \supset\!\text{I } 2\text{--}6 \\
8 \quad\left| \neg(A \supset \neg C) \supset (A \supset C) \right. & \supset\!\text{I } 1\text{--}7
\end{array}
$$

5.

1	J	want $J \vee (L \& \neg L)$
2	$J \vee (L \& \neg L)$	$\vee$I 1
3	$J \supset [J \vee (L \& \neg L)]$	$\supset$I 1–2
4	$J \vee (L \& \neg L)$	want J
5	$\neg J$	for reductio
6	$L \& \neg L$	$\vee$E 4, 5
7	L	$\&$E 6
8	$\neg L$	$\&$E 6
9	J	$\neg$E 5–7, 5–8
10	$(J \vee (L \& \neg L)) \supset J$	$\supset$I 4–9
11	$J \equiv [J \vee (L \& \neg L)]$	$\equiv$I 3, 10

Chapter 7 Part F

1.

1	$M \& (\neg N \supset \neg M)$	
2	M	$\&$E 1
3	$\neg N \supset \neg M$	$\&$E 1
4	$\neg\neg M$	DN 2
5	$\neg\neg N$	MT 3, 4
6	N	DN 5
7	$N \& M$	$\&$I 6, 2
8	$(N \& M) \vee \neg M$	$\vee$I 7

2.

1	$C \supset (E \& G)$	
2	$\neg C \supset G$	
3	$\neg G$	for reductio
4	$\neg\neg C$	MT 2, 3
5	C	DN 4
6	$E \& G$	$\supset$E 1, 5
7	G	& E 6
8	$\neg G$	R 3
9	G	$\neg$E 3–7, 3–8

3.

1	$(Z \& K) \equiv (Y \& M)$	
2	$D \& (D \supset M)$	
3	$D \supset M$	& E 2
4	D	& E 2
5	M	$\supset$E 3, 4
6	Y	want Z
7	$Y \& M$	& I 6, 5
8	$Z \& K$	$\equiv$E 1, 7
9	Z	& E 8
10	$Y \supset Z$	$\supset$I 6–9

4.

1	$(W \lor X) \lor (Y \lor Z)$	
2	$X \supset Y$	
3	$\neg Z$	
4	$\neg(W \lor Y)$	for reductio
5	$\neg W \,\&\, \neg Y$	DeM 4
6	$\neg Y$	& E 5
7	$\neg W$	& E 5
8	$\neg X$	MT 2, 6
9	$\neg W \,\&\, \neg X$	& I 7, 8
10	$\neg(W \lor X)$	DeM 9
11	$Y \lor Z$	$\lor$E 1, 10
12	Z	$\lor$E 11, 6
13	$\neg Z$	R 3
14	$W \lor Y$	$\neg$E 4–12, 4–13

Chapter 8

Chapter 8 Part B

1. $\exists x(Lpx \,\&\, Bx)$
2. $\exists x(Ldx \,\&\, Dx \,\&\, Wxb)$
3. $\exists x[Rcx \,\&\, Hx \,\&\, \exists y(By \,\&\, Lxy)]$
4. $\exists x(Bx \,\&\, Wxp \,\&\, [\exists y(Dy \,\&\, Lxy) \lor \exists y(Cy \,\&\, Lxy)])$
5. $\exists x(Bx \,\&\, Mx \,\&\, Wbx)$
6. $\exists x(\neg Bx \,\&\, Mx \,\&\, Wbx \,\&\, Lxd)$

Chapter 8 Part D

1. $Za \,\&\, Zb \,\&\, Zc$
2. $Rb \,\&\, \neg Ab$
3. $Lcb \supset Mb$
4. $(Ab \,\&\, Ac) \supset (Lab \,\&\, Lac)$
5. $\exists x(Rx \,\&\, Zx)$

6. $\forall x(Ax \supset Rx)$
7. $\forall x\big[Zx \supset (Mx \vee Ax)\big]$
8. $\exists x(Rx \,\&\, \neg Ax)$
9. $\exists x(Rx \,\&\, Lcx)$
10. $\forall x\big[(Mx \,\&\, Zx) \supset Lbx\big]$
11. $\forall x\big[(Mx \,\&\, Lax) \supset Lxa\big]$
12. $\exists xRx \supset Ra$
13. $\forall x(Ax \supset Rx)$
14. $\forall x\big[(Mx \,\&\, Lcx) \supset Lax\big]$
15. $\exists x(Mx \,\&\, Lxb \,\&\, \neg Lbx)$

Chapter 8 Part F

1. $Db \,\&\, Sb$
2. $Db \,\&\, De \,\&\, Df$
3. $Leb \,\&\, Lfe$
4. $\forall x(Dx \supset Sx)$
5. $\forall x(Sx \supset Dx)$
6. $\exists x(Dx \,\&\, Lxe)$
7. $\exists x(Dx \,\&\, Lxf) \supset \exists x(Dx \,\&\, Lxe)$
8. $\forall x(Lxe \supset \neg Sx)$
9. $\neg\exists x(Lxf \,\&\, Dx)$
10. $\forall x(\neg Sx \supset Lxb)$
11. $\exists x((Lxb \,\&\, Lex) \vee (Lxe \,\&\, Lbx))$
12. $\neg\exists x(Dx \,\&\, [(Lxb \,\&\, Lex) \vee (Lxe \,\&\, Lbx)])$
13. $\neg\exists x(Dx \,\&\, Lxx)$
14. $\forall x(Dx \supset \exists y(Dy \,\&\, Lyx))$
15. $\exists x(\forall y(Dy \supset Lyx))$

Chapter 8 Part H

1. $\neg\exists xTx$
2. $\forall x(Mx \supset Sx)$
3. $\exists x\neg Sx$
4. $\exists x[Cx \,\&\, \neg\exists yByx]$
5. $\neg\exists xBxx$
6. $\neg\exists x(Cx \,\&\, \neg Sx \,\&\, Tx)$
7. $\exists x(Cx \,\&\, Tx) \,\&\, \exists x(Mx \,\&\, Tx) \,\&\, \neg\exists x(Cx \,\&\, Mx \,\&\, Tx)$
8. $\forall x[Cx \supset \forall y(\neg Cy \supset Bxy)]$
9. $\forall x\big((Cx \,\&\, Mx) \supset \forall y[(\neg Cy \,\&\, \neg My) \supset Bxy]\big)$

Chapter 8 Part I

1. $\forall x(Fx \supset Tx)$
2. $\neg Rg \supset Tg$
3. $\forall x(Px \supset Lxg)$
4. $\exists x(Px \,\&\, Lxg) \supset Leg$
5. $\forall x((Fx \,\&\, Lfx) \supset Rx)$
6. $\neg\exists x(Px \,\&\, Lfx) \,\&\, \neg\exists y(Py \,\&\, Lyf)$
7. $\forall x((Px \,\&\, Lxg) \supset Lex)$
8. $\forall x\forall y((Px \,\&\, Py \,\&\, Lxy \,\&\, Ley) \supset Lex)$
9. $\exists x(Px \,\&\, Tx) \supset \forall y(Fy \supset Ry)$

Chapter 8 Part J

1. $\forall x(Cxp \supset Dx)$
2. $Cjp \,\&\, Fj$
3. $\exists x(Cxp \,\&\, Fx)$
4. $\neg\exists x Sxj$
5. $\forall x\big[(Cxp \,\&\, Fx) \supset Dx\big]$
6. $\neg\exists x(Cxp \,\&\, Mx)$
7. $\exists x(Cjx \,\&\, Sxe \,\&\, Fj)$
8. $Spe \,\&\, Mp$
9. $\forall x\big[(Sxp \,\&\, Mx) \supset \neg\exists y Cyx\big]$
10. $\exists x(Sxj \,\&\, \exists y Cyx \,\&\, Fj)$
11. $\forall x\big[Dx \supset \exists y(Sxy \,\&\, Fy \,\&\, Dy)\big]$
12. $\forall x\big[(Mx \,\&\, Dx) \supset \exists y(Cxy \,\&\, Dy)\big]$

Chapter 8 Part L

1. Rca, Rcb, Rcc, and Rcd are substitution instances of $\forall x Rcx$.
2. Of the expressions listed, only $\forall y Lby$ is a substitution instance of $\exists x\forall y Lxy$.

Chapter 9

Chapter 9 Part A

2, 3, 4, 6, 8, and 9 are true in the model.

Chapter 9 Part B

4, 5, and 7 are true in the model.

Chapter 9 Part C

1, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, and 15 are true in the model.

Chapter 9 Part D

$$UD = \{10,11,12,13\}$$
$$\text{extension}(O) = \{11,13\}$$
$$\text{extension}(S) = \varnothing$$
$$\text{extension}(T) = \{10,11,12,13\}$$
$$\text{extension}(U) = \{13\}$$
$$\text{extension}(N) = \{<11,10>,<12,11>,<13,12>\}$$

Chapter 9 Part E

1. The sentence is true in this model:
$$UD = \{\text{Stan}\}$$
$$\text{extension}(D) = \{\text{Stan}\}$$
$$\text{referent}(a) = \text{Stan}$$
$$\text{referent}(b) = \text{Stan}$$

   And it is false in this model:
$$UD = \{\text{Stan}\}$$
$$\text{extension}(D) = \varnothing$$
$$\text{referent}(a) = \text{Stan}$$
$$\text{referent}(b) = \text{Stan}$$

2. The sentence is true in this model:
$$UD = \{\text{Stan}\}$$
$$\text{extension}(T) = \{<\text{Stan, Stan}>\}$$
$$\text{referent}(h) = \text{Stan}$$

   And it is false in this model:
$$UD = \{\text{Stan}\}$$
$$\text{extension}(T) = \varnothing$$
$$\text{referent}(h) = \text{Stan}$$

3. The sentence is true in this model:
$$UD = \{\text{Stan, Ollie}\}$$
$$\text{extension}(P) = \{\text{Stan}\}$$
$$\text{referent}(m) = \text{Stan}$$

And it is false in this model:
$$UD = \{Stan\}$$
$$extension(P) = \varnothing$$
$$referent(m) = Stan$$

Chapter 9 Part F

There are many possible correct answers. Here are some:

1. Making the first sentence true and the second false:
$$UD = \{alpha\}$$
$$extension(J) = \{alpha\}$$
$$extension(K) = \varnothing$$
$$referent(a) = alpha$$

2. Making the first sentence true and the second false:
$$UD = \{alpha, omega\}$$
$$extension(J) = \{alpha\}$$
$$referent(m) = omega$$

3. Making the first sentence false and the second true:
$$UD = \{alpha, omega\}$$
$$extension(R) = \{<alpha, alpha>\}$$

4. Making the first sentence false and the second true:
$$UD = \{alpha, omega\}$$
$$extension(P) = \{alpha\}$$
$$extension(Q) = \varnothing$$
$$referent(c) = alpha$$

5. Making the first sentence true and the second false:
$$UD = \{iota\}$$
$$extension(P) = \varnothing$$
$$extension(Q) = \varnothing$$

6. Making the first sentence false and the second true:
$$UD = \{iota\}$$
$$extension(P) = \varnothing$$
$$extension(Q) = \{iota\}$$

7. Making the first sentence true and the second false:
$$UD = \{iota\}$$
$$extension(P) = \varnothing$$
$$extension(Q) = \{iota\}$$

8. Making the first sentence true and the second false:
$$UD = \{alpha, omega\}$$
$$extension(R) = \{<alpha, omega>, <omega, alpha>\}$$

9. Making the first sentence false and the second true:

$$\text{UD} = \{\text{alpha, omega}\}$$
$$\text{extension}(R) = \{<\text{alpha, alpha}>, <\text{alpha, omega}>\}$$

Chapter 9 Part I

2. No, it would not make any difference. The satisfaction of a formula with one or more free variables depends on what the variable assignment does for those variables. Because a sentence has no free variables, however, its satisfaction does not depend on the variable assignment. So a sentence that is satisfied by *some* variable assignment is satisfied by *every* other variable assignment as well.

Chapter 10

Chapter 10 Part A

1. $\forall x \forall y(Gxy \supset \exists z Gxz)$ is a tautology.

1.	$\neg\forall x\forall y(Gxy \supset \exists z Gxz)$ ✓a	
	$\vert$	
2.	$\neg\forall y(Gay \supset \exists z Gaz)$ ✓b	1 $\neg\forall$
3.	$\neg(Gab \supset \exists z Gaz)$ ✓	2 $\neg\forall$
	$\vert$	
4.	Gab	3 $\neg\supset$
5.	$\neg\exists z Gaz$ \b	
	$\vert$	
6.	$\neg Gab$	
	$\times$	
	4, 6	

2. $\forall x Fx \lor \forall x(Fx \supset Gx)$ is not a tautology.

1.	$\neg(\forall x Fx \lor \forall x(Fx \supset Gx))$ ✓	
	|	
2.	$\neg\forall x Fx$ ✓ a	1 $\neg\lor$
3.	$\neg\forall x(Fx \supset Gx)$ ✓ b	
	|	
4.	$\neg Fa$	2 $\neg\forall$
5.	$\neg(Fb \supset Gb)$ ✓	3 $\neg\forall$
	|	
6.	Fb	5 $\neg\supset$
7.	$\neg Gb$	
	↑	

UD={a, b}, extension(F)={b}, extension(G)=∅
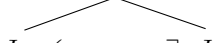
3. $\forall x(Fx \supset (\neg Fx \supset \forall y Gy))$ is a tautology.

1.	$\neg\forall x(Fx \supset (\neg Fx \supset \forall y Gy))$ ✓ a	
	|	
2.	$\neg(Fa \supset (\neg Fa \supset \forall y Gy))$ ✓	1 $\neg\forall$
	|	
3.	Fa	2 $\neg\supset$
4.	$\neg(\neg Fa \supset \forall y Gy)$ ✓	
	|	
5.	$\neg Fa$	4 $\neg\supset$
6.	$\neg\forall y Gy$	
	×	
	3, 5	

4. $\exists x(Fx \lor \neg Fx)$ is a tautology.

1.	$\neg\exists x(Fx \lor \neg Fx)$ \a	
	|	
2.	$\neg(Fa \lor \neg Fa)$ ✓	1 $\neg\exists$
	|	
3.	$\neg Fa$	2 $\neg\lor$
4.	$\neg\neg Fa$	
	×	
	3, 4	

5. $\exists x J x \equiv \neg \forall x \neg J x$ is a tautology.

1. $\neg(\exists x J x \equiv \neg \forall x \neg J x)$ ✓

2. $\exists x J x$ ✓a $\neg\exists x J x$ \a $1 \equiv$
3. $\neg\neg\forall x \neg J x$ ✓ $\neg\forall x \neg J x$ ✓a

4. $\neg\neg J a$ $3 \neg\forall$

5. $J a$ $2 \exists$

6. $\neg J a$ $2 \neg\exists$
                        $\times$
                       $4, 6$
7. $\forall x \neg J x$ \a $3 \neg\neg$

8. $\neg J a$ $7 \forall$
            $\times$
           $5, 8$

6. $\forall x(Fx \lor Gx) \supset (\forall y Fy \lor \exists x Gx)$ is a tautology.

1. $\neg(\forall x(Fx \lor Gx) \supset (\forall y Fy \lor \exists x Gx))$ ✓

2. $\forall x(Fx \lor Gx)$ \a $1 \neg\supset$
3. $\neg(\forall y Fy \lor \exists x Gx)$ ✓

4. $\neg\forall y Fy$ ✓a $3 \neg\lor$
5. $\neg\exists x Gx$ \a

6. $\neg Fa$ $4 \neg\forall$

7. $\neg Ga$ $5 \neg\exists$

8. $Fa \lor Ga$ ✓ $2 \forall$

9. Fa Ga $8 \lor$
                $\times$ $\times$

Chapter 10 Part B

1. Fa, Ga, $\therefore$ $\forall x(Fx \supset Gx)$ is invalid.

   1. Fa

   2. Ga

   3. $\neg\forall x(Fx \supset Gx)$ ✓b

                    |

   4. $\neg(Fb \supset Gb)$ ✓   3 $\neg\forall$

                    |

   5. Fb 4 $\neg\supset$

   6. $\neg Gb$

                 ↑

UD={a, b}, extension(F)={a, b}, extension(G)={a}

2. Fa, Ga, $\therefore$ $\exists x(Fx \,\&\, Gx)$ is valid.

   1. Fa

   2. Ga

   3. $\neg\exists x(Fx \,\&\, Gx)$ \\$a$

                    |

   4. $\neg(Fa \,\&\, Ga)$ ✓   3 $\neg\exists$

                 ⁀

   5. $\neg Fa$ $\neg Ga$ 4 $\neg\,\&$

               ×   ×

3. $\forall x \exists y Lxy$, $\therefore$ $\exists x \forall y Lxy$ is invalid.

1.	$\forall x \exists y Lxy$ \ $a, b, c, ...$	
2.	$\neg \exists x \forall y Lxy$ \ $a, b, c, ...$	
3.	$\exists y Lay$ ✓ b	1 $\forall$
4.	Lab	3 $\exists$
5.	$\neg \forall y Lay$ ✓ c	2 $\neg\exists$
6.	$\neg Lac$	5 $\neg\forall$
7.	$\exists y Lby$ ✓ d	1 $\forall$
8.	Lbd	7 $\exists$
9.	$\neg \forall y Lby$ ✓ e	2 $\neg\exists$
10.	$\neg Lbe$	9 $\neg\forall$
11.	$\exists y Lcy$ ✓ f	1 $\forall$
12.	Lcf	11 $\exists$
13.	$\neg \forall y Lcy$ ✓ g	2 $\neg\exists$
14.	$\neg Lcg$	13 $\neg\forall$
15.	$\vdots$	

This tree continues in an infinite way; each new name requires additional instances at line 1 and line 2, which in turn require two more new names. We can represent a model with an infinite UD, giving the extension of L as either a chart or an infinite set of ordered pairs. It is also possible to collapse this interpretation into a finite one; examining the chart, we can see that there is no reason the names couldn't be repeating names for the same two objects; we could let d and e be additional names for b and c, respectively, and so on for f and g, etc. So a two-object model will also suffice to satisfy the premises and falsify the conclusion, with UD={b, c} and extension(L)={<b, b>, <c, b>}. This is also shown in a second chart.

Lxy	a	b	c	d	e	f	g	...
a	-	1	0					
b				1	0			
c						1	0	
d							1	0
⋮								

Lxy	b	c
b	1	0
c	1	0

4. $\exists x(Fx \,\&\, Gx)$, $Fb \equiv Fa$, $Fc \supset Fa$, $\therefore Fa$ is invalid.

1.	$\exists x(Fx \,\&\, Gx)$ ✓d	
2.	$Fb \equiv Fa$ ✓	
3.	$Fc \supset Fa$ ✓	
4.	$\neg Fa$	
5.	$Fd \,\&\, Gd$ ✓	1 ∃
6.	Fd	5 &
7.	Gd	
8.	$Fb \qquad \neg Fb$	2 ≡
9.	$Fa \qquad \neg Fa$	
	×	
	4, 9	
10.	$\neg Fc \quad Fa$	3 ⊃
	↑ ×	
	9, 10	

UD={a, b, c, d}, extension(F)={d}, extension(G)={d}

5. $\forall x \exists y Gyx, \therefore \forall x \exists y (Gxy \lor Gyx)$ is valid.

1.	$\forall x \exists y Gyx \ \backslash a$	
2.	$\neg \forall x \exists y (Gxy \lor Gyx) \ \checkmark a$	
3.	$\neg \exists y (Gay \lor Gya) \ \backslash b$	2 $\neg \forall$
4.	$\exists y Gya \ \checkmark b$	1 $\forall$
5.	Gba	4 $\exists$
6.	$\neg (Gab \lor Gba) \ \checkmark$	3 $\neg \exists$
7.	$\neg Gab$	6 $\neg \lor$
8.	$\neg Gba$	
	$\times$	
	5, 8	

Chapter 11

Chapter 11 Part A

1. This system would not be sound. The satisfiability of an existential does not imply the satisfiability of any arbitrary instance. Here is a counterexample. This tree has a satisfiable root, but can close, given the proposed rule.

$\exists x Fx$
$\neg Fa$
$\quad |$
$\ Fa$
$\ \times$

2. This system would not be sound. Satisfying the existential doesn't guarantee satisfying its d instance. Here is a counterexample:

$\exists x Fx$
$\neg Fd$
$\quad |$
$\ Fd$
$\ \times$

3. This system would not be sound. The satisfiability of an existential does not imply the satisfiability of any three arbitrary instances. Here is a

counterexample. This tree has a satisfiable root, but can close, given the proposed rule.

$\exists x F x$
$\neg F a$
|
$F a$
$F b$
$F c$
$\times$

4. This system would still be sound; the proof extends. If we assume $\forall \chi \Phi$ is satisfiable, then some interpretation $\mathcal{I}$ satisfies it. This interpretation either satisfies all three instances (if the names were interpreted by $\mathcal{I}$), or it can easily be extended into a new interpretation $\mathcal{I}^*$ that satisfies them, by adding any new names and assigning them to any objects in $\mathcal{I}$'s UD, all of which we know satisfy Φ. So this rule will never take us from a satisfiable branch to an unsatisfiable one.

5. This system would still be sound; the proof extends. If we assume $\exists \chi \Phi$ is satisfiable, then some interpretation $\mathcal{I}$ satisfies it. Create a new interpretation $\mathcal{I}^*$, which includes the three names, and assign them each to an object in $\mathcal{I}$'s UD that satisfies Φ. We know there is at least one such object, since $\mathcal{I}(\exists \chi \Phi) = 1$. (Remember there is no prohibition on assigning multiple names to the same object.) $\mathcal{I}^*$ is guaranteed to satisfy the extension of the branch along with that which came above. So this rule will never take us from a satisfiable branch to an unsatisfiable one.

6. This system would still be sound; the proof extends. If we assume $\forall \chi \Phi$ is satisfiable, then some interpretation $\mathcal{I}$ satisfies it. Extend the interpretation to include the new name, and assign it to any object in the UD, which we know will satisfy Φ; the new interpretation satisfies the extension and what came before. So this rule will never take us from a satisfiable branch to an unsatisfiable one.

7. This system would still be sound; the proof extends. If χ does not occur in Φ, then $\exists \chi \Phi$ is logically equivalent to Φ. (Every substitution instance for χ of Φ will trivially just be Φ— since there is no χ in that sentence, replacing 'every' instance of that variable with any name results in no change at all.) So this rule is equivalent to the original conjunction rule, which is sound.

8. This system would still be sound. The soundness proof does not rely on the branch completion rules, so changing those rules will never interfere with the soundness proof.

9. This system would still be sound. The soundness proof does not rely on the branch completion rules, so changing those rules will never interfere with the soundness proof.

10. This system would remain sound. Adding an additional requirement for completion will never make it easier to close branches.

Chapter 11 Part B

1. This system would remain complete. If an existential is in a completed open branch, and this rule has been performed, then some instance of that existential is also in that branch. Any interpretation that satisfies that instance will also satisfy the existential.

2. This system would still be complete. If an existential is in a completed open branch, and this rule has been performed, then the d instance of that existential is also in that branch. Any interpretation that satisfies that instance will also satisfy the existential.

3. This system would still be complete. If an existential is in a completed open branch, and this rule has been performed, then three instances of that existential are also in that branch. Any interpretation that satisfies them will also satisfy the existential.

4. This system would not be complete. Satisfying three substitution instances for χ of Φ doesn't guarantee satisfying $\forall\chi\Phi$. Here is a counterexample to completeness— a tree with an unsatisfiable root that remains open.

   $\forall x Fx$ ✓
     $\neg Fa$
       |
     Fb
     Fc
     Fd

5. This system would still be complete; the proof extends. Satisfying substitution instances for χ of Φ— whether new or not, and no matter how many times— guarantees satisfying $\exists\chi\Phi$.

6. This system would not be complete. Satisfying an instance of a universal (whether or not the name is new) is no guarantee that the universal will be satisfied. Here is a counterexample:

   $\forall x Fx$ ✓
     $\neg Fa$
       |
     Fb

7. This system would still be complete; the proof extends. If χ does not occur in Φ, then $\exists\chi\Phi$ is logically equivalent to Φ. (Every substitution instance for χ of Φ will trivially just be Φ— since there is no χ in that sentence, replacing 'every' instance of that variable with any name results in no change at all.) So this rule is equivalent to the original conjunction rule, which is complete.

8. This system would not be complete. Taking one instance isn't enough to ensure that the universal is satisfied. Here is a counterexample:

$$\forall x F x \ \checkmark$$
$$\neg F a$$
$$|$$
$$F b$$

9. This system would not be complete. If a name is introduced later on in the tree and that instance of the universal hasn't been taken, satisfying the instances corresponding to the old names is not enough to guarantee satisfying the universal. Here is a counterexample to completeness:

1. Fa
2. $\forall x \forall y \neg Rxy \ \backslash a$
3. $\forall x \exists y Rxy \ \backslash a$
             $|$
4. $\forall y \neg Ray \ \backslash a$ $1 \ \forall$
5. $\exists y Ray \ \checkmark b$ $2 \ \forall$
             $|$
6. $\neg Raa$ $3 \ \neg\forall$
             $|$
7. Rab $4 \ \exists$

10. This system would remain complete. The addition is simply the new requirement that at least one new name be introduced via this rule; the reasoning that applied for completeness of the original system is unchanged. Satisfying every instance corresponding to a name in the branch will guarantee satisfying the general claims (the universals or negated existentials), since the UDs are constructed based on the names in the branch.

Chapter 12

Chapter 12 Part A

1. $Sh \ \& \ \neg \exists x (Vx \ \& \ Sx)$
2. $\neg Ki \supset \neg \exists x Kx$
3. $\neg \exists x (Sx \ \& \ Kx)$
4. $\neg Vh \ \& \ \neg Vi$
5. $\exists x (Vx \ \& \ Thx)$
6. $\forall x (Txi \supset \exists y (Txy \ \& \ Vy))$
7. $\forall x (Txi \supset \exists y (Txy \ \& \ \exists z (Tyz \ \& \ Vz)))$
8. $Ki \ \& \ \neg \exists x (Kx \ \& \ x \neq i)$
9. $Tih \ \& \ \neg \exists x (Tix \ \& \ x \neq h)$
10. $\exists x (Kx \ \& \ \neg \exists y (Ky \ \& \ x \neq y) \ \& \ Vx)$
11. $\exists x (Kx \ \& \ \neg \exists y (Ky \ \& \ x \neq y) \ \& \ \neg Sx)$

Chapter 12 Part B

1. $\forall x(Cx \supset Bx)$
2. $\neg\exists x Wx$
3. $\exists x\exists y(Cx \,\&\, Cy \,\&\, x{\neq}y)$
4. $\exists x\exists y(Jx \,\&\, Ox \,\&\, Jy \,\&\, Oy \,\&\, x{\neq}y)$
5. $\forall x\forall y\forall z\big[(Jx \,\&\, Ox \,\&\, Jy \,\&\, Oy \,\&\, Jz \,\&\, Oz) \supset (x{=}y \vee x{=}z \vee y{=}z)\big]$
6. $\exists x\exists y\big(Jx \,\&\, Bx \,\&\, Jy \,\&\, By \,\&\, x{\neq}y \,\&\, \forall z[(Jz \,\&\, Bz) \supset (x{=}z \vee y{=}z)]\big)$
7. $\exists x_1\exists x_2\exists x_3\exists x_4(Dx_1 \,\&\, Dx_2 \,\&\, Dx_3 \,\&\, Dx_4 \,\&\, x_1{\neq}x_2 \,\&\, x_1{\neq}x_3 \,\&\, x_1{\neq}x_4 \,\&\, x_2{\neq}x_3 \,\&\, x_2{\neq}x_4 \,\&\, x_3{\neq}x_4$
   $\&\, \neg\exists y(Dy \,\&\, y{\neq}x_1 \,\&\, y{\neq}x_2 \,\&\, y{\neq}x_3 \,\&\, y{\neq}x_4))$
8. $\exists x\big(Dx \,\&\, Cx \,\&\, \forall y[(Dy \,\&\, Cy) \supset x{=}y] \,\&\, Bx\big)$
9. $\forall x\big[(Ox \,\&\, Jx) \supset Wx\big] \,\&\, \exists x\big[Mx \,\&\, \forall y(My \supset x{=}y) \,\&\, Wx\big]$
10. $\exists x\big(Dx \,\&\, Cx \,\&\, \forall y[(Dy \,\&\, Cy) \supset x{=}y] \,\&\, Wx\big) \supset \exists x(Wx \,\&\, \forall y(Wy \supset x{=}y))$
11. $\exists x\big[Mx \,\&\, \forall y(My \supset x{=}y) \,\&\, \neg Jx\big]$
12. $\exists x\exists z\big(Dx \,\&\, Cx \,\&\, Mz \,\&\, \forall y[(Dy \,\&\, Cy) \supset x{=}y] \,\&\, \forall y[(My \supset z{=}y) \,\&\, x{\neq}z]\big)$

Chapter 12 Part C

1. $\exists x(Bbx \,\&\, Bwx)$
2. $\forall x\forall y[\exists z(Gz \,\&\, Sx \,\&\, Sy \,\&\, Bxz \,\&\, Byz) \supset x{=}y]$
3. $\exists x(Sx \,\&\, x{\neq}b \,\&\, Fx)$
4. $\forall x[(Fx \,\&\, \neg Wx) \supset Awx]$
5. $\forall x(x{\neq}f \supset Kfx)$
6. $\forall x(\exists y(Sy \,\&\, Axy \,\&\, \neg\exists z((Vz \vee Dz) \,\&\, Kxz)) \supset Kbx)$
7. $\forall x(Gx \supset \exists y(Sy \,\&\, Byx))$
8. $\forall x(Gx \supset \exists y(Sy \,\&\, Byx \,\&\, \forall z[(Vz \vee Dz \vee Fz) \supset Ayz]))$
9. $\forall x(Gx \supset \exists y(Sy \,\&\, Byx \,\&$
   $\forall z[(Vz \vee Dz \vee Fz) \supset Ayz] \,\&$
   $\forall y_2(\forall z[(Vz \vee Dz \vee Fz) \supset Ay_2z] \supset y_2{=}y)))$

Chapter 12 Part E

1. There are many possible answers. Here is one:

   $$\text{UD} = \{\text{Harry, Sally}\}$$
   $$\text{extension}(R) = \{<\text{Sally, Harry}>\}$$
   $$\text{referent}(a) = \text{Harry}$$

2. There are no predicates or constants, so we only need to give a UD. Any UD with 2 members will do.

3. We need to show that it is impossible to construct a model in which these are both true. Suppose $\exists x\ x{\neq}a$ is true in a model. There is something in the universe of discourse that is *not* the referent of a. So there are at least two things in the universe of discourse: referent(a) and this other thing.

Call this other thing β— we know $a{\neq}\beta$. But if $a{\neq}\beta$, then $\forall x \forall y\ x{=}y$ is false. So the first sentence must be false if the second sentence is true. As such, there is no model in which they are both true. Therefore, they are inconsistent.

4. A model where this is true:

$$\text{UD} = \{\text{Harry}\}$$
$$\text{referent}(h) = \text{Harry}$$
$$\text{referent}(i) = \text{Harry}$$

A model where this is false:

$$\text{UD} = \{\text{Harry, Ivan}\}$$
$$\text{referent}(h) = \text{Harry}$$
$$\text{referent}(i) = \text{Ivan}$$

5. This is one of many possible answers:

$$\text{UD} = \{\text{Dorothy, Harry}\}$$
$$\text{extension}(Z) = \{<\text{Harry}>\}$$
$$\text{referent}(d) = \text{Dorothy}$$
$$\text{referent}(h) = \text{Harry}$$
$$\text{referent}(s) = \text{Dorothy}$$

6. This is one of a number of possible answers. Any model with a 1 object UD that satisfies the premise will work.

$$\text{UD} = \{\text{Dorothy}\}$$
$$\text{extension}(D) = \{\}$$
$$\text{extension}(T) = \{\}$$
$$\text{referent}(d) = \text{Dorothy}$$

Chapter 12 Part F

1. $\models \forall x \forall y (x{=}y \supset y{=}x)$ is true.

1.	$\neg\forall x\forall y(x{=}y \supset y{=}x)$ ✓ a	
	|	
2.	$\neg\forall y(a{=}y \supset y{=}a)$ ✓ b	1 $\neg\forall$
3.	$\neg(a{=}b \supset b{=}a)$ ✓	2 $\neg\forall$
	|	
4.	$a{=}b$	3 $\neg\supset$
5.	$b{\neq}a$	
	|	
6.	$b{\neq}b$	4, 5 $=$
	$\times$	
	6	

2. $\models \forall x \exists y\ x{=}y$ is true.

    1. $\neg \forall x \exists y\ x{=}y\ \checkmark a$

                 |

    2. $\neg \exists y\ y{=}a\ \backslash a$ $1\ \neg\forall$

                 |

    3. $a{\neq}a$ $2\ \neg\exists$
                    $\times$
                    3

3. $\models \exists x \forall y\ x{=}y$ is false.

    1. $\neg \exists x \forall y\ x{=}y\ \backslash a, b$

                 |

    2. $\neg \forall y\ a{=}y\ \checkmark b$ $1\ \neg\exists$

                 |

    3. $a{\neq}b$ $2\ \neg\forall$

                 |

    4. $\neg \forall y\ b{=}y\ \checkmark c$ $1\ \neg\exists$

                 |

    5. $b{\neq}c$ $4\ \neg\forall$

                 |

    6. $\vdots$

    UD $= \{$a, b, c, $\ldots\}$

This infinite tree describes a model with an infinite UD. Since there are no predicates we needn't provide any extensions. Note that a two-object domain would also have sufficed to satisfy the root, even though the tree was not complete after line 3, since we hadn't taken the b instance of the negated existential. The tree will be infinite since at line 6 we would take the c instance from line 1, etc.

4. $\exists x \forall y \; x{=}y \models \forall x \forall y (Rxy \equiv Ryx)$ is true.

1.	$\exists x \forall y \; x{=}y$ ✓a	
2.	$\neg \forall x \forall y (Rxy \equiv Ryx)$ ✓b	
	$\mid$	
3.	$\forall y \; a{=}y$ \b, c	1 $\exists$
	$\mid$	
4.	$\neg \forall y (Rby \equiv Ryb)$ ✓c	2 $\neg\forall$
5.	$\neg (Rbc \equiv Rcb)$ ✓	4 $\neg\forall$
	$\mid$	
6.	$a{=}b$	3 $\forall$
7.	$a{=}c$	3 $\forall$
	$\mid$	
8.	$\neg (Rac \equiv Rca)$	5, 6 $=$
9.	$\neg (Raa \equiv Raa)$ ✓	5, 7 $=$
10.	$Raa \qquad Raa$	9 $\neg\equiv$
11.	$\neg Raa \quad \neg Raa$	
	$\times \qquad\; \times$	
	10, 11 $\;\;$ 10, 11	

5. $\models \neg \forall x \forall y \forall z [(Axy \,\&\, Azx \,\&\, y{=}z) \supset Axx]$ is false.
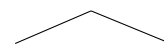
1.	$\neg\neg\forall x \forall y \forall z ((Axy \,\&\, Azx \,\&\, y{=}z) \supset Axx)$ ✓	
	$\mid$	
2.	$\forall x \forall y \forall z ((Axy \,\&\, Azx \,\&\, y{=}z) \supset Axx)$ \$a$	1 $\neg\neg$
3.	$\forall y \forall z ((Aay \,\&\, Aza \,\&\, y{=}z) \supset Aaa)$ \$a$	2 $\forall$
4.	$\forall z ((Aaa \,\&\, Aza \,\&\, a{=}z) \supset Aaa)$ \$a$	3 $\forall$
5.	$(Aaa \,\&\, Aaa \,\&\, a{=}a) \supset Aaa$ ✓	4 $\forall$
6.	$\neg (Aaa \,\&\, Aaa \,\&\, a{=}a)$ ✓ $\qquad Aaa$	5 $\supset$
	$\qquad\qquad\qquad\qquad\qquad\qquad\uparrow$	
7.	$\neg Aaa \quad \neg Aaa \quad a{\neq}a$	6 $\neg\&$
	$\uparrow \qquad\; \uparrow \qquad\;\; \times$	
	7	

This tree is complete. Three open branches describe these two models:

UD = {a}	UD = {a}
extension(A) = ∅	extension(A) = {<a, a>}

6. $\forall x \forall y\ x{=}y \models \exists x F x \equiv \forall x F x$ is true.

1.	$\forall x \forall y\ x{=}y\ \backslash a$	
2.	$\neg(\exists x F x \equiv \forall x F x)\ \checkmark$	

	3.	$\exists x F x\ \checkmark a$	$\neg\exists x F x\ \backslash a$	$2 \equiv$
	4.	$\neg\forall x F x\ \checkmark b$	$\forall x F x\ \backslash a$	
	5.		$\neg F a$	$3\ \neg\exists$
	6.		$F a$	$4\ \forall$
			$\times$	
	7.	$F a$		$3\ \exists$
	8.	$\neg F b$		$4\ \neg\forall$
	9.	$\forall y\ a{=}y\ \backslash b$		$1\ \forall$
	10.	$a{=}b$		$9\ \forall$
	11.	$\neg F a$		$10, 8 =$
		$\times$		
		7, 11		

7. $\forall x(x{=}a \ \lor \ x{=}b), Ga \ \equiv \ \neg Gb \ \models \ \neg\exists x\exists y\exists z(Gx \,\&\, Gy \,\&\, \neg Gz)$ is false.

1.	$\forall x(x{=}a \lor x{=}b) \ \backslash c, d, e, a, b$	
2.	$Ga \equiv \neg Gb \ \checkmark$	
3.	$\neg\neg\exists x\exists y\exists z(Gx \,\&\, Gy \,\&\, \neg Gz) \ \checkmark$	
4.	$\exists x\exists y\exists z(Gx \,\&\, Gy \,\&\, \neg Gz) \ \checkmark c$	$3 \ \neg\neg$
5.	$\exists y\exists z(Gc \,\&\, Gy \,\&\, \neg Gz) \ \checkmark d$	$4 \ \exists$
6.	$\exists z(Gx \,\&\, Gd \,\&\, \neg Gz) \ \checkmark e$	$5 \ \exists$
7.	$Gc \,\&\, Gd \,\&\, \neg Ge \ \checkmark$	
8.	Gc	$7 \ \&$
9.	Gd	
10.	$\neg Ge$	
11.	$c{=}a \lor c{=}b \ \checkmark$	$1 \ \forall$
12.	$c{=}a \qquad c{=}b$	$11 \ \lor$
13.	Ga	$8, 12 \ =$
14.	$Ga \qquad\qquad \neg Ga$	$2 \ \equiv$
15.	$\neg Gb \qquad\qquad Gb$	
	$\qquad\qquad\qquad \times$	
16.	$d{=}a \lor d{=}b \ \checkmark \qquad {}^{13,\,14}$	$1 \ \forall$
17.	$d{=}a \quad d{=}b$	$16 \ \lor$
18.	$e{=}a \lor e{=}b \ \checkmark$	$1 \ \forall$
19.	$e{=}a \qquad\qquad e{=}b$	$18 \ \lor$
20.	$\neg Ga \qquad a{=}a \lor a{=}b \ \checkmark$	$1 \ \forall$
	$\times$	
21.	$^{13,\,20} \qquad a{=}a \quad a{=}b$	$20 \ \lor$
22.	$b{=}a \lor b{=}b \ \checkmark$	$1 \ \forall$
23.	$b{=}a \quad b{=}b$	$22 \ \lor$
	$\qquad\quad \uparrow$	
24.	Gb	
	$\times$	
	${}^{15,\,24}$	

This tree will end up with multiple open branches; since one open branch is enough to falsify the entailment claim, this version focuses on the left-most

open branch at every point, leaving the other open branches incomplete. It describes this model:

$$\text{UD} = \{a, b\}$$
$$\text{referent}(a) = a$$
$$\text{referent}(b) = b$$
$$\text{referent}(c) = a$$
$$\text{referent}(d) = a$$
$$\text{referent}(e) = b$$
$$\text{extension}(G) = \{a\}$$

8. $\forall x(Fx \supset x{=}f), \exists x(Fx \vee \forall y\, y{=}x) \models Ff$ is false.

1.	$\forall x(Fx \supset x{=}f)$ \a, f	
2.	$\exists x(Fx \vee \forall y\, y{=}a)$ ✓ a	
3.	$\neg Ff$	
	│	
4.	$Fa \vee \forall y\, y{=}a$ ✓	2 ∃
	│	
5.	$Fa \supset a{=}f$ ✓	1 ∀
6.	$\neg Fa \quad a{=}f$	5 ⊃
7.	$Fa \quad \forall y\, y{=}a$ \f, a	4 ∨
	×	
8.	$f{=}a$	7 ∀
9.	$a{=}a$	7 ∀
	│	
10.	$Ff \supset f{=}f$ ✓	1 ∀
11.	$\neg Ff \quad f{=}f$	
	│ ↑	
12.	$\neg Fa$	$8, 11 =$
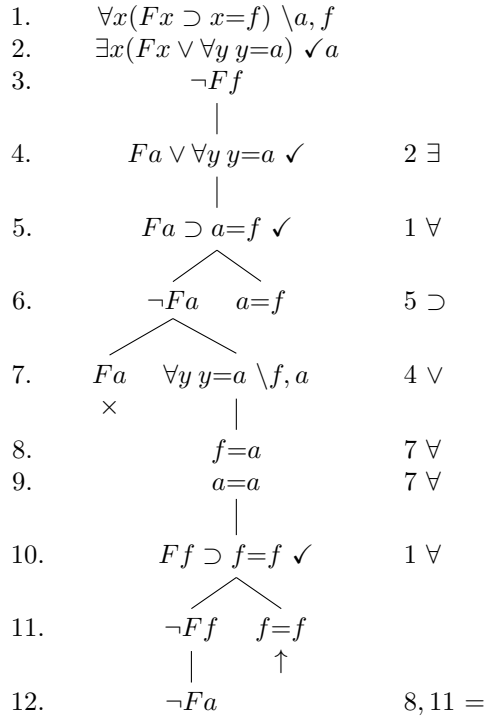
We have one completed open branch. (Another open branch has not been completed.) It gives this model:

$$\text{UD} = \{a\}$$
$$\text{referent}(a) = a$$
$$\text{referent}(f) = a$$
$$\text{extension}(F) = \varnothing$$

9. $\exists x \exists y Dxy \models \forall x_1 \forall x_2 \forall x_3 \forall x_4[(Dx_1x_2 \,\&\, Dx_3x_4) \supset (x_2 {\neq} x_3 \lor Dx_1x_4)]$ is false.

1.	$\exists x \exists y Dxy$ ✓a	
2.	$\neg \forall x_1 \forall x_2 \forall x_3 \forall x_4((Dx_1x_2 \,\&\, Dx_3x_4) \supset (x_2 {\neq} x_3 \lor Dx_1x_4))$ ✓c	
3.	$\exists y Day$ ✓b	$1\ \exists$
4.	Dab	$3\ \exists$
5.	$\neg \forall x_2 \forall x_3 \forall x_4((Dcx_2 \,\&\, Dx_3x_4) \supset (x_2 {\neq} x_3 \lor Dcx_4))$ ✓d	$2\ \neg\forall$
6.	$\neg \forall x_3 \forall x_4((Dcd \,\&\, Dx_3x_4) \supset (d {\neq} x_3 \lor Dcx_4))$ ✓e	$5\ \neg\forall$
7.	$\neg \forall x_4((Dcd \,\&\, Dex_4) \supset (d {\neq} e \lor Dcx_4))$ ✓f	$6\ \neg\forall$
8.	$\neg((Dcd \,\&\, Def) \supset (d {\neq} e \lor Dcf))$ ✓	$7\ \neg\forall$
9.	$Dcd \,\&\, Def$ ✓	$8\ \neg\supset$
10.	$\neg(d {\neq} e \lor Dcf)$ ✓	$10\ \neg\supset$
11.	Dcd	$9\ \&$
12.	Def	
13.	$\neg d {\neq} e$	$10\ \neg\supset$
14.	$\neg Dcf$	
15.	$d{=}e$	$13\ \neg\neg$
16.	Ddf	$12, 15\ =$
17.	Dce	$11, 15\ =$
	↑	

$$\text{UD} = \{\text{a, b, c, d, f}\}$$

referent(a) = a
referent(b) = b
referent(c) = c
referent(d) = d
referent(e) = d
referent(f) = f

extension(D) =

Dxy	a	b	c	d	f
a	-	1	-	-	-
b	-	-	-	-	-
c	-	-	-	1	0
d	-	-	-	-	1
f	-	-	-	-	-

Chapter 13

Chapter 13 Part A

1	$\forall x \exists y (Rxy \lor Ryx)$	
2	$\forall x \neg Rmx$	
3	$\exists y (Rmy \lor Rym)$	$\forall$E 1
4	$\quad Rma \lor Ram$	for $\exists$E
5	$\quad \neg Rma$	$\forall$E 2
6	$\quad Ram$	$\lor$E 4, 5
7	$\quad \exists x Rxm$	$\exists$I 6
8	$\exists x Rxm$	$\exists$E 3, 4–7

1	$\forall x (\exists y Lxy \supset \forall z Lzx)$	
2	$\quad Lab$	
3	$\quad \exists y Lay \supset \forall z Lza$	$\forall$E 1
4	$\quad \exists y Lay$	$\exists$I 2
5	$\quad \forall z Lza$	$\supset$E 3, 4
6	$\quad Lca$	$\forall$E 5
7	$\quad \exists y Lcy \supset \forall z Lzc$	$\forall$E 1
8	$\quad \exists y Lcy$	$\exists$I 6
9	$\quad \forall z Lzc$	$\supset$E 7, 8
10	$\quad Lcc$	$\forall$E 9
11	$\quad \forall x Lxx$	$\forall$I 10

1	$\forall x(Jx \supset Kx)$	
2	$\exists x \forall y Lxy$	
3	$\forall x Jx$	
4	$\forall y Lay$	for $\exists$E
5	Ja	$\forall$E 3
6	$Ja \supset Ka$	$\forall$E 1
7	Ka	$\supset$E 6, 5
8	Laa	$\forall$E 4
9	$Ka \& Laa$	$\&$I 7, 8
10	$\exists x(Kx \& Lxx)$	$\exists$I 9
11	$\exists x(Kx \& Lxx)$	$\exists$E 2, 4–10

1	$\neg(\exists x Mx \vee \forall x \neg Mx)$	for reductio
2	$\neg\exists x Mx \& \neg\forall x \neg Mx$	DeM 1
3	$\neg\exists x Mx$	$\&$E 2
4	$\forall x \neg Mx$	QN 3
5	$\neg\forall x \neg Mx$	$\&$E 2
6	$\exists x Mx \vee \forall x \neg Mx$	$\neg$E 1–4, 1–5

Chapter 13 Part B

1.

1	$\neg(\forall x Fx \vee \neg\forall x Fx)$	for reductio
2	$\neg\forall x Fx \& \neg\neg\forall x Fx$	DeM 1
3	$\neg\forall x Fx$	$\&$E 2
4	$\neg\neg\forall x Fx$	$\&$E 2
5	$\forall x Fx \vee \neg\forall x Fx$	$\neg$E 1–3, 1–4

2.

1	$\forall x(Mx \equiv Nx)$	
2	$Ma \,\&\, \exists x Rxa$	want $\exists x Nx$
3	$Ma \equiv Na$	$\forall$E 1
4	Ma	$\&$ E 2
5	Na	$\equiv$E 3, 4
6	$\exists x Nx$	$\exists$I 5

3.

1	$\forall x(\neg Mx \vee Ljx)$	
2	$\forall x(Bx \supset Ljx)$	
3	$\forall x(Mx \vee Bx)$	want $\forall x Ljx$
4	$\neg Ma \vee Lja$	$\forall$E 1
5	$Ma \supset Lja$	MC 4
6	$Ba \supset Lja$	$\forall$E 2
7	$Ma \vee Ba$	$\forall$E 3
8	Lja	DIL 7, 5, 6
9	$\forall x Ljx$	$\forall$I 8

4.

1	$\forall x(Cx \,\&\, Dt)$	want $\forall x Cx \,\&\, Dt$
2	$Ca \,\&\, Dt$	$\forall$E 1
3	Ca	$\&$ E 2
4	$\forall x Cx$	$\forall$I 3
5	Dt	$\&$ E 2
6	$\forall x Cx \,\&\, Dt$	$\&$ I 4, 5

5.

1	$\exists x(Cx \lor Dt)$	want $\exists xCx \lor Dt$
2	$Ca \lor Dt$	for $\exists$E
3	$\neg(\exists xCx \lor Dt)$	for reductio
4	$\neg\exists xCx \,\&\, \neg Dt$	DeM 3
5	$\neg Dt$	$\&$ E 4
6	Ca	$\lor$E 2, 5
7	$\exists xCx$	$\exists$I 6
8	$\neg\exists xCx$	$\&$ E 4
9	$\exists xCx \lor Dt$	$\neg$E 3–7, 3–8
10	$\exists xCx \lor Dt$	$\exists$E 1, 2–9

Chapter 13 Part E

1.

1	$\forall x\forall yGxy$	
2	$\forall yGay$	$\forall$E 1
3	Gaa	$\forall$E 2
4	$\exists xGxx$	$\exists$I 3

2.

1	$\forall x\forall y(Gxy \supset Gyx)$	
2	$\forall y(Gay \supset Gya)$	$\forall$E 1
3	$Gab \supset Gba$	$\forall$E 2
4	$\forall y(Gby \supset Gyb)$	$\forall$E 1
5	$Gba \supset Gab$	$\forall$E 4
6	$Gab \equiv Gba$	$\equiv$I 3, 5
7	$\forall y(Gay \equiv Gya)$	$\forall$I 6
8	$\forall x\forall y(Gxy \equiv Gyx)$	$\forall$I 7

3.

1	$\forall x(Ax \supset Bx)$	
2	$\exists xAx$	want $\exists xBx$
3	Aa	for $\exists$E
4	$Aa \supset Ba$	$\forall$E 1
5	Ba	$\supset$E 3, 4
6	$\exists xBx$	$\exists$I 5
7	$\exists xBx$	$\exists$E 2, 3–6

4.

1	$Na \supset \forall x(Mx \equiv Ma)$	
2	Ma	
3	$\neg Mb$	want $\neg Na$
4	Na	for reductio
5	$\forall x(Mx \equiv Ma)$	$\supset$E 1, 4
6	$Mb \equiv Ma$	$\forall$E 5
7	Mb	$\equiv$E 2, 6
8	$\neg Mb$	R 3
9	$\neg Na$	$\neg$I 4–7, 4–8

5.

1	Pa	want Pa
2	Pa	R 1
3	$Pa \supset Pa$	$\supset$I 1–2
4	$\neg Pa \vee Pa$	MC 3
5	$Pa \vee \neg Pa$	Comm 4
6	$\forall z(Pz \vee \neg Pz)$	$\forall$I 5

6.

1	$\forall xRxx$	want $\supset \exists x\exists yRxy$
2	Raa	$\forall$E 1
3	$\exists yRay$	$\exists$I 2
4	$\exists x\exists yRxy$	$\exists$I 3
5	$\forall xRxx \supset \exists x\exists yRxy$	$\supset$I 1–4

7.

1	Qa	want Qa
2	Qa	R 1
3	$Qa \supset Qa$	$\supset$I 1–2
4	$\exists x(Qa \supset Qx)$	$\exists$I 3
5	$\forall y \exists x(Qy \supset Qx)$	$\forall$I 4

Chapter 13 Part H

Regarding the translation of this argument, see p. .

1	$\exists x \forall y[\forall z(Lxz \supset Lyz) \supset Lxy]$	
2	$\forall y[\forall z(Laz \supset Lyz) \supset Lay]$	for $\exists$E
3	$\forall z(Laz \supset Laz) \supset Laa$	$\forall$E 2
4	$\neg \exists x Lxx$	for reductio
5	$\forall x \neg Lxx$	QN 4
6	$\neg Laa$	$\forall$E 5
7	$\neg \forall z(Laz \supset Laz)$	MT 3, 6
8	Lab	want Lab
9	Lab	R 8
10	$Lab \supset Lab$	$\supset$I 8–9
11	$\forall z(Laz \supset Laz)$	$\forall$I 10
12	$\neg \forall z(Laz \supset Laz)$	R 7
13	$\exists x Lxx$	$\neg$E 4–12
14	$\exists x Lxx$	$\exists$E 1, 2–13

Chapter 13 Part J

2, 3, and 5 are logically equivalent.

Chapter 13 Part K

2, 4, 5, 7, and 10 are valid. Here are complete answers for some of them:

1.   UD = {mocha, freddo}
      extension(R) = {<mocha, freddo>, <freddo, mocha>}

2.

1	$\exists y \forall x Rxy$	want $\forall x \exists y Rxy$
2	$\forall x Rxa$	for $\exists$E
3	Rba	$\forall$E 2
4	$\exists y Rby$	$\exists$I 3
5	$\forall x \exists y Rxy$	$\forall$I 4
6	$\forall x \exists y Rxy$	$\exists$E 1, 2–5

Appendix C

Quick Reference

Characteristic Truth Tables

Φ	$\neg\Phi$
1	0
0	1

Φ	Ψ	$\Phi\,\&\,\Psi$	$\Phi\vee\Psi$	$\Phi\supset\Psi$	$\Phi\equiv\Psi$
1	1	1	1	1	1
1	0	0	1	0	0
0	1	0	1	1	0
0	0	0	0	1	1

Symbolization

SENTENTIAL CONNECTIVES (chapter 2)

It is not the case that P.	$\neg P$
Either P, or Q.	$(P \vee Q)$
Neither P, nor Q.	$\neg(P \vee Q)$ or $(\neg P \,\&\, \neg Q)$
Both P, and Q.	$(P \,\&\, Q)$
If P, then Q.	$(P \supset Q)$
P only if Q.	$(P \supset Q)$
P if and only if Q.	$(P \equiv Q)$
Unless P, Q. P unless Q.	$(P \vee Q)$

PREDICATES (chapter 8)

All Fs are Gs.	$\forall x(Fx \supset Gx)$
Some Fs are Gs.	$\exists x(Fx \,\&\, Gx)$
Not all Fs are Gs.	$\neg\forall x(Fx \supset Gx)$ or $\exists x(Fx \,\&\, \neg Gx)$
No Fs are Gs.	$\forall x(Fx \supset \neg Gx)$ or $\neg\exists x(Fx \,\&\, Gx)$

IDENTITY (chapter 12)

Only j is G.	$\forall x(Gx \equiv x{=}j)$
Everything besides j is G.	$\forall x(x{\neq}j \supset Gx)$
The F is G.	$\exists x(Fx \,\&\, \forall y(Fy \supset x{=}y) \,\&\, Gx)$
The F is not G.	$\exists x(Fx \,\&\, \forall y(Fy \supset x{=}y) \,\&\, \neg Gx)$

Tree Rules

Sentential Rules (chapter 5)

$\Phi \mathbin{\&} \Psi$
|
Φ
Ψ

$\Phi \equiv \Psi$
∧
Φ $\neg\Phi$
Ψ $\neg\Psi$

$\Phi \supset \Psi$
∧
$\neg\Phi$ Ψ

$\Phi \vee \Psi$
∧
Φ Ψ

$\neg\neg\Phi$
|
Φ

$\neg(\Phi \vee \Psi)$
|
$\neg\Phi$
$\neg\Psi$

$\neg(\Phi \mathbin{\&} \Psi)$
∧
$\neg\Phi$ $\neg\Psi$

$\neg(\Phi \equiv \Psi)$
∧
Φ $\neg\Phi$
$\neg\Psi$ Ψ

$\neg(\Phi \supset \Psi)$
|
Φ
$\neg\Psi$
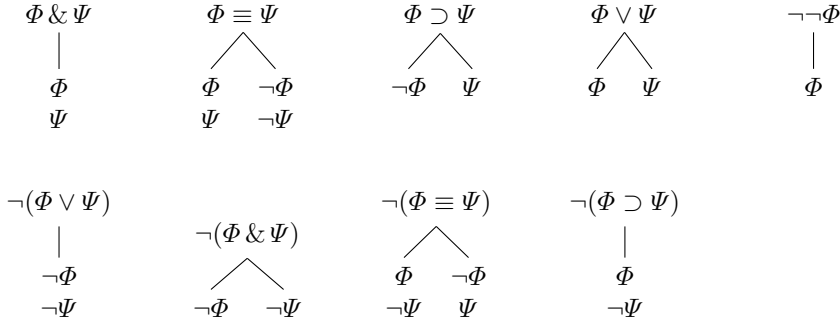
Quantifier Rules (chapter 8)

$\exists \chi \Phi \ \checkmark a$
|
$\Phi \boxed{\chi \Rightarrow a}$

*a must be new

$\forall \chi \Phi \ \backslash a$
|
$\Phi \boxed{\chi \Rightarrow a}$

for any a

$\neg\exists \chi \Phi \ \backslash a$
|
$\neg\Phi \boxed{\chi \Rightarrow a}$

for any a

$\neg\forall \chi \Phi \ \checkmark a$
|
$\neg\Phi \boxed{\chi \Rightarrow a}$

*a must be new
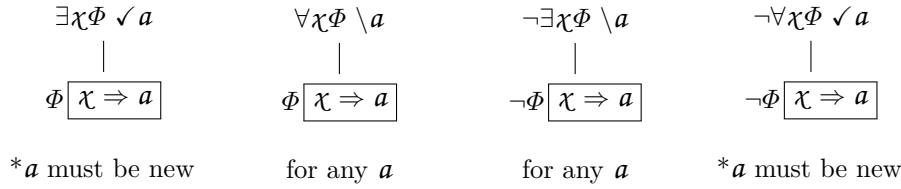
Identity Rules (chapter 12)

If a branch contains an identity claim $a{=}b$ and some sentence Φ, then you may add $\Phi \boxed{a \Rightarrow b}$ or $\Phi \boxed{b \Rightarrow a}$ to the branch.

A branch is COMPLETE if and only if either (i) it is closed, or (ii) all of the following conditions are met:

1. Every resolvable sentence in the branch has been resolved;

2. For every general quantified sentence Φ and every name a in the branch, the a instance of Φ is in the branch; and

3. For every identity claim $a{=}b$ in the branch, either:

   (a) for every atomic sentence or negated atomic sentence Φ containing a in the branch, $\Phi \boxed{a \Rightarrow b}$ is in the branch, or

   (b) for every atomic sentence or negated atomic sentence Φ containing b in the branch, $\Phi \boxed{b \Rightarrow a}$ is in the branch.

A branch closes if and only if either (i) it contains some sentence Φ along with its negation, $\neg\Phi$, or (ii) it contains, for some name a, the sentence $a{\neq}a$.

Natural Deduction Rules

Basic Rules

Reiteration

$$
\begin{array}{c|l}
m & \Phi \\[2pt]
  & \Phi \qquad \text{R } m
\end{array}
$$

Conjunction Introduction

$$
\begin{array}{c|l}
m & \Phi \\[2pt]
n & \Psi \\[2pt]
  & \Phi \,\&\, \Psi \qquad \&\text{I } m,\, n
\end{array}
$$

Conjunction Elimination

$$
\begin{array}{c|l}
m & \Phi \,\&\, \Psi \\[2pt]
  & \Phi \qquad \&\text{E } m \\[2pt]
  & \Psi \qquad \&\text{E } m
\end{array}
$$

Disjunction Introduction

$$
\begin{array}{c|l}
m & \Phi \\[2pt]
  & \Phi \lor \Psi \qquad \lor\text{I } m \\[2pt]
  & \Psi \lor \Phi \qquad \lor\text{I } m
\end{array}
$$

Disjunction Elimination

$$
\begin{array}{c|l}
m & \Phi \lor \Psi \\[2pt]
n & \neg\Psi \\[2pt]
  & \Phi \qquad \lor\text{E } m,\, n
\end{array}
$$

$$
\begin{array}{c|l}
m & \Phi \lor \Psi \\[2pt]
n & \neg\Phi \\[2pt]
  & \Psi \qquad \lor\text{E } m,\, n
\end{array}
$$

Conditional Introduction

$$
\begin{array}{c|ll}
m & \;\;\Phi & \text{want } \Psi \\[2pt]
n & \;\;\Psi & \\[2pt]
  & \Phi \supset \Psi & \supset\text{I } m\text{–}n
\end{array}
$$

Conditional Elimination

$$
\begin{array}{c|l}
m & \Phi \supset \Psi \\[2pt]
n & \Phi \\[2pt]
  & \Psi \qquad \supset\text{E } m,\, n
\end{array}
$$

Biconditional Introduction

$$
\begin{array}{c|l}
m & \Phi \supset \Psi \\[2pt]
n & \Psi \supset \Phi \\[2pt]
  & \Phi \equiv \Psi \qquad \equiv\text{I } m,\, n
\end{array}
$$

Biconditional Elimination

$$
\begin{array}{c|l}
m & \Phi \equiv \Psi \\[2pt]
n & \Psi \\[2pt]
  & \Phi \qquad \equiv\text{E } m,\, n
\end{array}
$$

$$
\begin{array}{c|l}
m & \Phi \equiv \Psi \\[2pt]
n & \Phi \\[2pt]
  & \Psi \qquad \equiv\text{E } m,\, n
\end{array}
$$

Negation Introduction

$$
\begin{array}{c|ll}
m & \;\;\Phi & \text{for reductio} \\[2pt]
n & \;\;\Psi & \\[2pt]
o & \;\;\neg\Psi & \\[2pt]
p & \neg\Phi & \neg\text{I } m\text{–}n,\, m\text{–}o
\end{array}
$$

Negation Elimination

$$
\begin{array}{c|ll}
m & \;\;\neg\Phi & \text{for reductio} \\[2pt]
n & \;\;\Psi & \\[2pt]
o & \;\;\neg\Psi & \\[2pt]
p & \Phi & \neg\text{E } m\text{–}n,\, m\text{–}o
\end{array}
$$

Quantifier Rules

EXISTENTIAL INTRODUCTION

$$m \quad \Phi$$
$$\exists \chi \Phi \boxed{\boxed{c^* \Rightarrow \chi}} \qquad \exists \text{I } m$$

* χ *may replace some or all occurrences of* c.

EXISTENTIAL ELIMINATION

$$m \quad \exists \chi \Phi$$
$$n \quad \quad \Phi \boxed{\chi \Rightarrow c^*}$$
$$p \quad \quad \Psi$$
$$\Psi \qquad \exists \text{E } m, n\text{–}p$$

* c *must not appear outside the subproof.*

UNIVERSAL INTRODUCTION

$$m \quad \Phi$$
$$\forall \chi \Phi \boxed{c^* \Rightarrow \chi} \qquad \forall \text{I } m$$

* c *must not occur in any undischarged assumptions.*

UNIVERSAL ELIMINATION

$$m \quad \forall \chi \Phi$$
$$\Phi \boxed{\chi \Rightarrow c} \qquad \forall \text{E } m$$

Identity Rules

$$c{=}c \qquad =\text{I}$$

$$m \quad c{=}d$$
$$n \quad \Phi$$
$$\Phi \boxed{\boxed{c \Rightarrow d}} \qquad =\text{E } m, n$$
$$\Phi \boxed{\boxed{d \Rightarrow c}} \qquad =\text{E } m, n$$

One constant may replace some or all occurrences of the other.

Derived Rules

DILEMMA

$$m \quad \Phi \vee \Psi$$
$$n \quad \Phi \supset \Omega$$
$$p \quad \Psi \supset \Omega$$
$$\Omega \qquad \text{DIL } m, n, p$$

MODUS TOLLENS

$$m \quad \Phi \supset \Psi$$
$$n \quad \neg \Psi$$
$$\neg \Phi \qquad \text{MT } m, n$$

HYPOTHETICAL SYLLOGISM

$$m \quad \Phi \supset \Psi$$
$$n \quad \Psi \supset \Omega$$
$$\Phi \supset \Omega \qquad \text{HS } m, n$$

Replacement Rules

(Replacemnet rules may be used within complex sentences or on whole lines.)

COMMUTATIVITY (Comm)
$$(\Phi \,\&\, \Psi) \Longleftrightarrow (\Psi \,\&\, \Phi)$$
$$(\Phi \vee \Psi) \Longleftrightarrow (\Psi \vee \Phi)$$
$$(\Phi \equiv \Psi) \Longleftrightarrow (\Psi \equiv \Phi)$$

DEMORGAN (DeM)
$$\neg(\Phi \vee \Psi) \Longleftrightarrow (\neg \Phi \,\&\, \neg \Psi)$$
$$\neg(\Phi \,\&\, \Psi) \Longleftrightarrow (\neg \Phi \vee \neg \Psi)$$

DOUBLE NEGATION (DN)
$$\neg\neg \Phi \Longleftrightarrow \Phi$$

MATERIAL CONDITIONAL (MC)
$$(\Phi \supset \Psi) \Longleftrightarrow (\neg \Phi \vee \Psi)$$
$$(\Phi \vee \Psi) \Longleftrightarrow (\neg \Phi \supset \Psi)$$

BICONDITIONAL EXCHANGE ($\equiv$ex)
$$[(\Phi \supset \Psi) \,\&\, (\Psi \supset \Phi)] \Longleftrightarrow (\Phi \equiv \Psi)$$

QUANTIFIER NEGATION (QN)
$$\neg \forall \chi \Phi \Longleftrightarrow \exists \chi \neg \Phi$$
$$\neg \exists \chi \Phi \Longleftrightarrow \forall \chi \neg \Phi$$

In the Introduction to his volume *Symbolic Logic*, Charles Lutwidge Dodson advised: "When you come to any passage you don't understand, *read it again*: if you *still* don't understand it, *read it again*: if you fail, even after *three* readings, very likely your brain is getting a little tired. In that case, put the book away, and take to other occupations, and next day, when you come to it fresh, you will very likely find that it is *quite* easy."

The same might be said for this volume, although readers are forgiven if they take a break for snacks after *two* readings.

about the authors:
P.D. Magnus is an associate professor of philosophy in Albany, New York. His primary research is in the philosophy of science.

Jonathan Ichikawa is an associate professor of philosophy in Vancouver, British Columbia. His primary research is in epistemology.