forallχ in *Lurch*

An Introduction to Formal Logic in *Lurch*

P.D. Magnus
University at Albany, State University of New York
Nathan Carter
Bentley University

http://web.bentley.edu/empl/c/ncarter/faxil, version 1.1 [171002]

In his original text, P.D. Magnus thanked the people who made his 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, and the students of Introduction to Logic, who detected various errors in previous versions of the book.

Nathan Carter thanks P.D. for creating the text in the first place and Ken Monks for being the other half of the *Lurch* project. Thanks also go to Bentley University for supporting me for a summer of working on this textbook, and to all my logic students, who help shape my teaching and the development of *Lurch*.

Typesetting was carried out entirely in LaTeX2ε. The style for typesetting proofs is based on fitch.sty (v0.4) by Peter Selinger, University of Ottawa.

This copy of `forall`χ in *Lurch* is current as of October 2, 2017. The most recent version is available on-line at `http://web.bentley.edu/empl/c/ncarter/faxil`

Contents

Preface

For many years, I (Nathan Carter) used the text `forall`χ by P.D. Magnus in my Introduction to Mathematical Logic course for honors students at Bentley University. The text was well-written, clear, at the right level, and free on the Internet—perfect! (The original is still online at `fecundity.com/logic` for those interested.)

I also work on the mathematical software project *Lurch* with Kenneth G. Monks of the University of Scranton, and in the years preceding 2013 that project matured significantly, to the point where it can help students out with just about all the homework I assign them in that logic course. Thus it was time to integrate *Lurch* into `forall`χ.

There are three benefits to doing so. First, I get the perfect text for my own course, because I want to teach logic using both `forall`χ and *Lurch*. Second, any other instructor interested in using *Lurch* in an introduction to logic course would have an easy way to do so, because I would distribute my textbook online for free, under the same license that Magnus distributed the original. Instructors could just adopt it as their text for the course and be ready to go. Third, the *Lurch* project would benefit from having an easy way for new users to adopt the software. Thus this book was born!

The changes from `forall`χ to this text can be summarized as follows.

1. Material on how to use *Lurch* was added to the text.
2. A less formal style of proof-writing was adopted. (Though tight requirements on what remains a correct proof remain, they are built into the software, and not enforced as specific typographical requirements.)
3. Sections of the text were rearranged to suit my teaching preferences.
4. Five chapters were added to the end of the text, to show the application of logic to mathematics, and teach the transition from formal proofs to more typical proof-writing. These chapters are optional for a reader who is interested only in logic, and not in its application to mathematics.

Chapter 1

Getting started with *Lurch*

1.1 What is *Lurch*?

Lurch is a mathematical word processor that can check your reasoning. This will be very handy in the homework for this course, where you will be doing a lot of formal reasoning. Although it can't help with every single aspect of this course, it will help with some of the most difficult parts.

In this chapter, we will just see the very basics of *Lurch*, including how to install it and type a simple document into it. Then you'll be ready to use it for the first homework problems in Chapter 2.

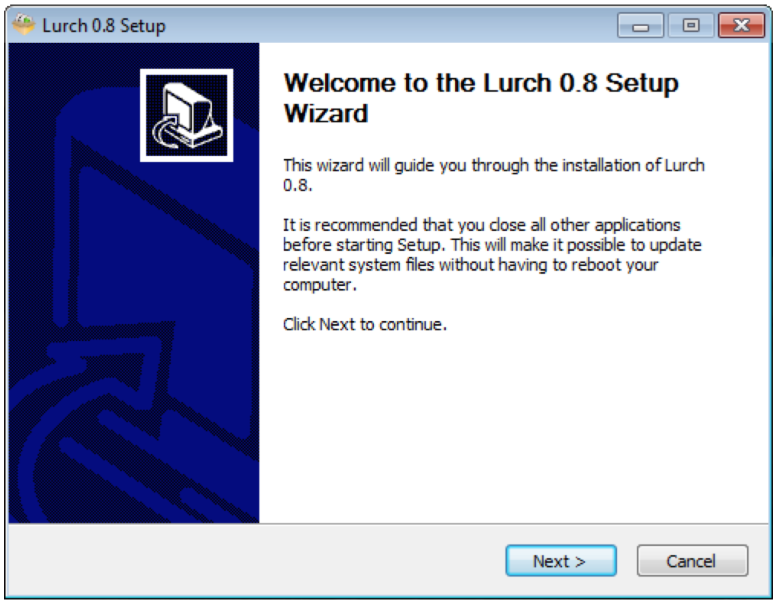
Thus this chapter is very easy; it's all basic computer use! Next chapter we start learning logic.

1.2 Installing the software

To install *Lurch* on your computer, proceed to its website, `lurchmath.org`, and click the Download link on the top of the page. The resulting page should tell you the current version of the software, and give you a large download link that looks like Figure 1.1. Click that link.

On the next page, your download should start in a few seconds. Once it completes, you will need to find where your browser placed the downloaded file. This varies from browser to browser, and I expect that you have downloaded files from the Internet before and can find the one you just downloaded. How you use that

Figure 1.1: The download link on the *Lurch* website


Lurch 0.8 Setup

Welcome to the Lurch 0.8 Setup Wizard

This wizard will guide you through the installation of Lurch 0.8.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 1.2: The first screen of the *Lurch* installer on Windows

file depends on your platform, and I address each of the three major platforms here.

Windows

The downloaded file for Windows users is an executable installer with the extension `.exe`. Double-click it to run it. It may require you to grant it permission to install software on your system, and you should do so. The first screen of the installation proceedure will look something like Figure 1.2.

Most users will want to accept the installer's default options and just click 'Next' enough times to complete the installation. But if you have preferences about where software is installed on your system, feel free to customize the installation. After it completes, the installer will run *Lurch*.

In the future, when you want to launch *Lurch*, use the shortcut on your desktop or in the Start Menu. For now, proceed to Section 1.3 to read about using the software for the first time.

Mac OS X

The downloaded file for Mac users is a disk image with the extension `.dmg`. Double-click the disk image to mount it. OS X should then show you the contents of the disk image, which are the *Lurch* application and a link to your computer's Applications folder. It should look like Figure 1.3.

Click and drag *Lurch* into your Applications folder as that figure suggests. When the copying is done, you can close the disk image, eject it, and drag the disk image to the Trash. You can run *Lurch* by navigating to your Applications folder and double-clicking the *Lurch* application. Do so now, and then proceed to Section 1.3 to read about using the software for the first time.

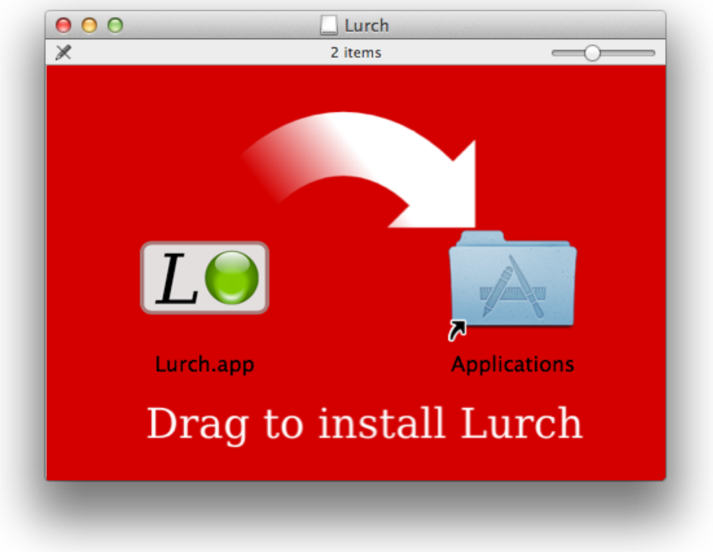


Figure 1.3: The contents of the *Lurch* disk image on Mac OS X

Linux

The downloaded file for Linux users is a bash script. Linux users of *Lurch* must run this file from the terminal, and may need to give it executable privileges first. The script ensures that your system has all the necessary packages and libraries installed, downloads the *Lurch* source code from the Internet, and builds and installs *Lurch* on your system.

This has been tested on a few different Linux distributions, but Linux users who encounter difficulty should contact the *Lurch* community by the *Lurch* email list, linked to from the website.

1.3 A math word processor

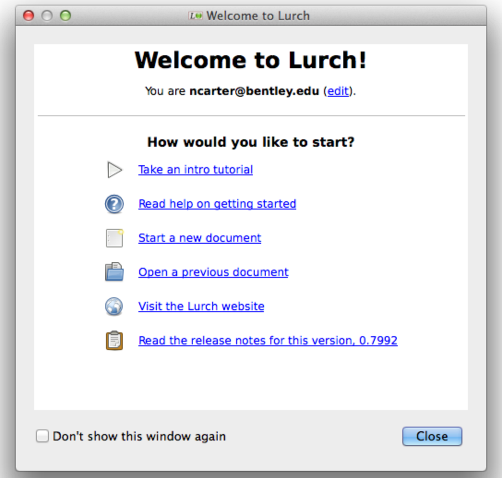
Throughout the remainder of this text, all images of the *Lurch* user interface will be shown on Mac OS X, only because that is the author's chosen platform. *Lurch* runs well on Windows and Linux also.

The introductory window

When *Lurch* first opens, it shows an introductory window, as in Figure 1.4. New users will have the word 'Anonymous' instead of their email address, and should use the 'edit' link to enter their own name or email. Once *Lurch* knows who you are, it can put your name on documents that you author.

This introductory window will always launch when *Lurch* opens unless you tell it not to, with the checkbox on the bottom left. If you banish this window for good, but want it back later, open the *Lurch* preferences and check the box called 'Show introductory dialog at startup.'

Close the window once you have specified your name. We may return to the introductory tutorial later, but we're going to start even simpler than that.


Welcome to Lurch

Welcome to Lurch!

You are **ncarter@bentley.edu** (edit).

How would you like to start?

Take an intro tutorial

Read help on getting started

Start a new document

Open a previous document

Visit the Lurch website

Read the release notes for this version, 0.7992

Don't show this window again

Close

Figure 1.4: The introductory dialog shown on the first run of the *Lurch* application

A *Lurch* document

Once you've closed the introductory dialog, you'll be presented with a blank document, and some typical word-processor toolbars above it, something like Figure 1.5. With your cursor in this window, you can type text, just like you can in any word processor.

Feel free to experiment with the following controls shown on the toolbars and in the menus, to verify that they behave as you expect.

1. Alignment (left, right, center, justified)
2. Outline (indent, unindent, bullets, numbers)
3. Fonts (font name, font size, styles, bold, italic, underline, colors)
4. Hyperlinks (add, edit, remove)

You should also find some tools that you've never seen before, such as buttons for 'meaningful expressions,' 'properties,' 'contexts,' and 'validation.' We'll leave these tools alone for now, but they form an essential part of *Lurch* that we'll use a lot in future chapters.

Saving your work

When you want to save a document, proceed as you would in any software: Choose 'Save' from the File menu. You may wish to create a folder for this course, and save all your *Lurch* files in the same folder.

Lurch saves documents with a `.lurch` extension (rather than `.txt` or `.rtf` or `.doc`, as in other word processors). Such `.lurch` files can only be opened by *Lurch*, using the 'Open' item on the File menu.

To share a *Lurch* document with someone who doesn't have *Lurch*, you have three choices.

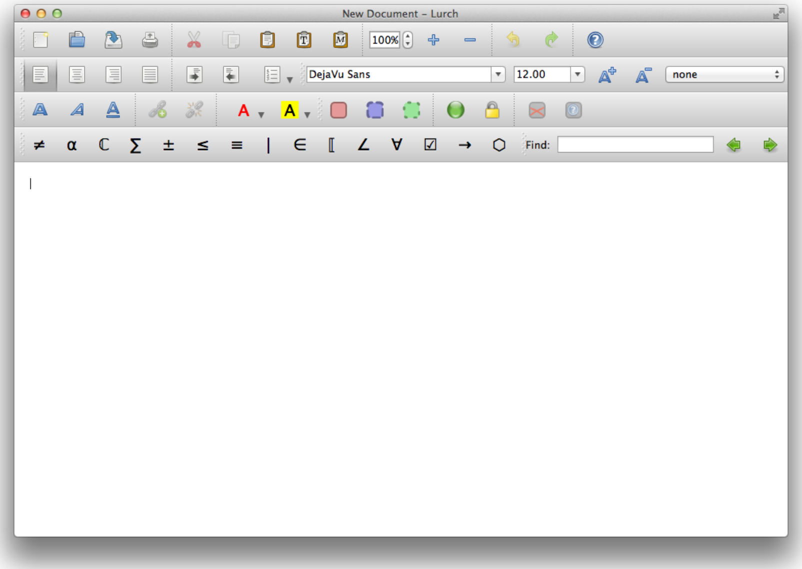


Figure 1.5: A new, blank *Lurch* document with the cursor ready to type in it

1. With simple *Lurch* documents, you can simply copy and paste your text from *Lurch* into an email message or other document.

   When we start using the advanced features of *Lurch*, there will be document content that cannot survive a copy-and-paste operation. Then you will need to use one of the other two options below, which preserve your document exactly as you see it in *Lurch*.

2. Ask *Lurch* to print your document, and use the the print-to-PDF feature that many modern platforms provide. Then share the PDF, which can be opened without *Lurch*.

3. From the File menu, choose 'Save as webpage' to create a self-contained `.html` file. This file can also be read by anyone in their web browser, or you can paste its contents directly into an email.

1.4 Beyond word processing

In future chapters, we will see how to type math and logic symbols into *Lurch*, and even get it to give us feedback on our logical reasoning. *Lurch* works like a free tutor, and is an invaluable tool when learning the more complex mathematics in this text. But for now, just get used to the *Lurch* interface so that you're ready to use it to type your Chapter 2 homework.

Chapter 2

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.

In logic, we are not interested in the teeth-gnashing, hair-pulling kind of argument. A logical argument is structured to give someone a reason to believe some conclusion. Here is one such argument:

(1) It is raining heavily.
(2) If you do not take an umbrella, you will get soaked.
$\therefore$ You should take an umbrella.

The three dots on the third line of the 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.

2.1 Arguments

When people mean to give arguments, they often use words like 'therefore' and 'because.' When analyzing an argument, the first thing to do is to separate the premises from the conclusion. Words like these are a clue to what the argument is supposed to be, especially if— in the argument as given— the conclusion comes at the beginning or in the middle of the argument.

premise indicators: since, because, given that

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

To be perfectly general, we can define 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. If the premises are true and the argument is a good one, then you have a reason to accept the conclusion.

Notice that this definition is quite general. Consider this example:

There is coffee in the coffee pot.

There is a dragon playing bassoon on the armoire.

∴ Salvador Dali was a poker player.

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. Nevertheless, given our definition, it still counts as an argument— albeit a bad one.

2.2 Sentences

In logic, we are only interested in sentences that can figure as a premise or conclusion of an argument. So we will say that a SENTENCE is something that can be true or false.

You should not 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.'

Also, there are things that would count as 'sentences' in a linguistics or grammar course that we will not count as sentences in logic.

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. For this reason, questions will not count as sentences in logic. 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 'Ouch!' is sometimes called an exclamatory sentence, but it is 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.

2.3 Two ways that arguments can go wrong

Consider the argument that you should take an umbrella (on p. 12, above). If premise (1) is false— if it is sunny outside— then the argument gives you no reason to carry an umbrella. Even if it is raining outside, you might not need an umbrella. You might wear a rain poncho or keep to covered walkways. In these cases, premise (2) would be false, since you could go out without an umbrella and still avoid getting soaked.

Suppose for a moment that both the premises are true. You do not own a rain poncho. You need to go places where there are no covered walkways. Now does the argument show you that you should take an umbrella? Not necessarily. Perhaps you enjoy walking in the rain, and you would like to get soaked. In that case, even though the premises were true, the conclusion would be false.

For any argument, there are two ways that it 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. Even if the premises were true, the form of the argument might be weak. The example we just considered is weak in both ways.

When an argument is weak in the second way, there is something wrong with the *logical form* of the argument: Premises of the kind given do not necessarily lead to a conclusion of the kind given. We will be interested primarily in the logical form of arguments.

Consider another example:

You are reading this book.

This is a logic book.

∴ You are a logic student.

This is not a terrible argument. Most people who read this book are logic students. Yet, it is possible for someone besides a logic student to read this book. If your roommate picked up the book and thumbed through it, they would not immediately become a logic student. So the premises of this argument, even though they are true, do not guarantee the truth of the conclusion. Its logical form is less than perfect.

An argument that had no weakness of the second kind would have perfect logical form. If its premises were true, then its conclusion would *necessarily* be true. We call such an argument 'deductively valid' or just 'valid.'

Even though we might count the argument above as a good argument in some sense, it is not valid; that is, it is 'invalid.' One important task of logic is to sort valid arguments from invalid arguments.

Quiz Yourself

- Give two example English sentences that do not count as sentences in the sense introduced in this chapter.
- What are the two ways that arguments can go wrong?
- If an argument's premises being true leads to its conclusion also being true just over 50% of the time, is the argument valid or invalid?

2.4 Deductive validity

An argument is deductively VALID if and only if it is impossible for the premises to be true and the conclusion false.

The crucial thing about a valid argument is that it is impossible for the premises to be true *at the same time* that the conclusion is false. Keep in mind that validity is about the form of an argument, not about whether its premises or conclusion are true by themselves. Validity is about the relationship among the statements. Consider this example:

Oranges are either fruits or musical instruments.
Oranges are not fruits.
∴ Oranges are musical instruments.

The conclusion of this argument is ridiculous. Nevertheless, it follows validly from the premises. This is a valid argument. *If* both premises were true (which, in this unusual case, requires some imagination), *then* the conclusion would necessarily be true.

This shows that a deductively valid argument does not need to have true premises or a true conclusion. Conversely, having true premises and a true conclusion is not enough to make an argument valid. Consider this example:

London is in England.
Beijing is in China.
∴ Paris is in France.

The premises and conclusion of this argument are, as a matter of fact, all true. This is a terrible argument, however, because the premises have nothing to do with the conclusion. Imagine what would happen if Paris declared independence from the rest of France. Then the conclusion would be false, even though the premises would both still be true. Thus, it is *logically possible* for the premises of this argument to be true and the conclusion false. The argument is invalid.

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

Inductive arguments

There can be good arguments which nevertheless fail to be deductively valid. Consider this one:

In January 1997, it rained in San Diego.
In January 1998, it rained in San Diego.
In January 1999, it rained in San Diego.
∴ It rains every January in San Diego.

This is an INDUCTIVE argument, because it generalizes from many cases to a conclusion about all cases.

Certainly, the argument could be made stronger by adding additional premises: In January 2000, it rained in San Diego. In January 2001... and so on. Regardless of how many premises we add, however, the argument will still not be deductively valid. It is possible, although unlikely, that it will fail to rain next January in San Diego. Moreover, we know that the weather can be fickle. No amount of evidence should convince us that it rains there *every* January. Who is to say that some year will not be a freakish year in which there is no rain in January in San Diego; even a single counter-example is enough to make the conclusion of the argument false.

Inductive arguments, even good inductive arguments, are not deductively valid. We will not be interested in inductive arguments in this book.

2.5 Other logical notions

In addition to deductive validity, we will be interested in some other logical concepts.

Truth values

True or false is said to be the TRUTH VALUE of a sentence. We defined sentences as things that could be true or false; we could have said instead that sentences are things that can have truth values.

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. Yet there are some sentences that must be true, just as a matter of logic.

Consider these sentences:

1. It is raining.
2. Either it is raining, or it is not.
3. It is both raining and not raining.

In order to know if sentence 1 is true, you would need to look outside or check the weather channel. Logically speaking, it might be either true or false. Sentences like this are called *contingent* sentences.

Sentence 2 is different. You do not need to look outside to know that it is true. Regardless of what the weather is like, it is either raining or not. This sentence is *logically true*; it is true merely as a matter of logic, regardless of what the world is actually like. A logically true sentence is called 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. It might be raining here and not raining across town, it might be raining now but stop raining even as you read this, but it is impossible for it to be both raining and not raining here at this moment. The third sentence is *logically false*; it is false regardless of what the world is like. A logically false sentence is called a CONTRADICTION.

To be precise, we can define a CONTINGENT SENTENCE as a sentence that is neither a tautology nor a contradiction.

A sentence might *always* be true and still be contingent. For instance, if there never were a time when the universe contained fewer than seven things, then the sentence ‘At least seven things exist’ would always be true. Yet the sentence is contingent; its truth is not a matter of logic. There is no contradiction in imagining a world in which there are fewer than seven things. The important question is whether the sentence *must* be true, just on account of logic.

Logical equivalence

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

John went to the store after he washed the dishes.
John washed the dishes before he went to the store.

These two sentences are both contingent, since John might not have gone to the store or washed dishes at all. 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.

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 and 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 the inconsistency has nothing at all to do with G2. G2 just happens to be part of an inconsistent set.

Sometimes, people will say that an inconsistent set of sentences 'contains a contradiction.' By this, they do not necessarily mean that one of the sentences is a contradiction on its own. Rather, they mean that it would be logically impossible for all of the sentences to be true at once. A set might be inconsistent even if each of the sentences in it is either contingent or tautologous.

Summary of logical notions

▷ 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.

Quiz Yourself

- Is an inductive argument deductively valid?
- For an argument to be valid, do the premises and conclusion all need to be true?
- What do we call a sentence that is sometimes true and sometimes false?
- When we speak of logical equivalence, how many sentences are under consideration?

2.6 Formal languages

Here is a famous valid argument:

Socrates is a man.
All men are mortal.
∴ Socrates is mortal.

This is an iron-clad argument. The only way you could challenge the conclusion is by denying one of the premises— the logical form is impeccable. What about this next argument?

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

This argument might be less interesting than the first, because the second premise is obviously false. There is no clear sense in which all men are carrots. Yet the argument is valid. To see this, notice that both arguments have this form:

S is M.
All Ms are Cs.
∴ S is C.

In both arguments S stands for Socrates and M stands for man. In the first argument, C stands for mortal; in the second, C stands for carrot. Both arguments have this form, and every argument of this form is valid. So both arguments are valid.

What we did here was replace words like 'man' or 'carrot' with symbols like 'M' or 'C' so as to make the logical form explicit. *This is the central idea behind formal logic.* We want to remove irrelevant or distracting features of the argument to make the logical form more obvious.

Starting with an argument in a *natural language* like English, we translate the argument into a *formal language*. Parts of the English sentences are replaced with letters and symbols. The goal is to reveal the formal structure of the argument, as we did with these two.

There are formal languages that work like the symbolization we gave for these two arguments. A logical system like this was developed by Aristotle, a philosopher who lived in Greece during the 4th century BC. Aristotle was a student of Plato and the tutor of Alexander the Great. Aristotle's logic, with some revisions, was the dominant logic in the western world for more than two millennia.

In Aristotelean logic, categories are replaced with capital letters. Every sentence of an argument is then represented as having one of four forms, which medieval logicians labeled in this way: (A) All As are Bs. (E) No As are Bs. (I) Some A is B. (O) Some A is not B.

It is then possible to describe valid *syllogisms*, three-line arguments like the two we considered above. Medieval logicians gave mnemonic names to all of the valid argument forms. The form of our two arguments, for instance, was called *Barbara*. The vowels in the name, all As, represent the fact that the two premises and the conclusion are all (A) form sentences.

There are many limitations to Aristotelean logic. One is that it makes no distinction between kinds and individuals. So the first premise might just as well be written 'All Ss are Ms': All Socrateses are men. Despite its historical importance, Aristotelean logic has been superceded. The remainder of this book will develop two formal languages, and a logical system corresponding to each.

The first is SL, which stands for *sentential logic*. In SL, the smallest units are sentences themselves. Simple sentences are represented as letters and connected with logical connectives like 'and' and 'not' to make more complex sentences.

The second is QL, which stands for *quantified logic*. In QL, the basic units are objects, properties of objects, and relations between objects.

When we translate an argument into a formal language, we hope to make its logical structure clearer. We want to include enough of the structure of the English language argument so that we can judge whether the argument is valid or invalid. If we included every feature of the English language, all of the subtlety and nuance, then there would be no advantage in translating to a formal language. We might as well think about the argument in English.

At the same time, we would like a formal language that allows us to represent many kinds of English language arguments. This is one reason to prefer QL to Aristotelean logic; QL can represent every valid argument of Aristotelean logic and more.

So when deciding on a formal language, there is inevitably a tension between wanting to capture as much structure as possible and wanting a simple formal language— simpler formal languages leave out more. This means that 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*. Aristotelean logic, SL, and QL are all bivalent, but there are limits to the power of bivalent logic. For instance, some philosophers have claimed that the future is not yet determined. If they are right, then sentences about *what will be the case* are not yet true or false. Some formal languages accommodate this by allowing for sentences that are neither true nor false, but something in between. Other formal languages, so-called paraconsistent logics, allow for sentences that are both true *and* false.

The languages presented in this book are not the only possible formal languages. However, most nonstandard logics extend on the basic formal structure of the bivalent logics discussed in this book. So this is a good place to start.

Quiz Yourself

- What is the central idea behind formal logic?
- What are the smallest units of sentential logic?
- What does it mean for the systems in this book to be bivalent?

2.7 Typing arguments in *Lurch*

I encourage you to get used to *Lurch* by typing every homework assignment in it, despite the fact that you don't yet know how to get *Lurch* to check your work. Doing so will familiarize you with the software while the work is still easier, so that you'll already be comfortable when the material gets harder.

You may have noticed that *Lurch* has several math symbols in the bottom toolbar in Figure 1.5 on page 11. Clicking any one of them opens a pane of related symbols, which you can click to insert into your document.

For instance, if you were typing the argument from page 12 into *Lurch*, you would need the mathematical symbol $\therefore$, which is on the toolbar pane labeled with the $\forall$ symbol. Try typing that argument into *Lurch* now, resulting in a document like the one in Figure 2.1.

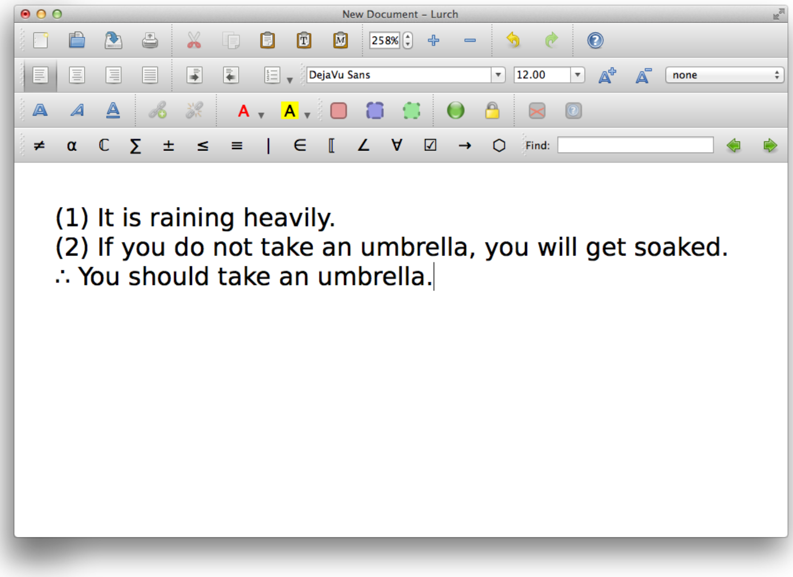
Since you will type mathematical symbols often, *Lurch* provides shortcuts for entering each symbol. To find out what they are, click on one of the toolbar buttons (such as the one marked with the $\forall$ symbol) and hover your mouse over one of its symbols, as shown in Figure 2.2. The yellow box that shows up teaches you three things: the symbol's name (in this case 'therefore'), its keyboard shortcut (in this case `\therefore`), and its Unicode value (not relevant here).

So to enter $\therefore$ into a document without using your mouse, just type the keyboard shortcut (in this case, a backslash \ followed by the word `therefore`). When you then press the spacebar, that shorcut will be replaced by the symbol. Because each symbol has a mnemonic name, you can quickly learn many such shortcuts, and enter long sequences of symbols comparatively quickly.

Of course, users who prefer the mouse are always free to use it. Take this opportunity to do the following homework assignment in *Lurch* to familiarize yourself with its interface.

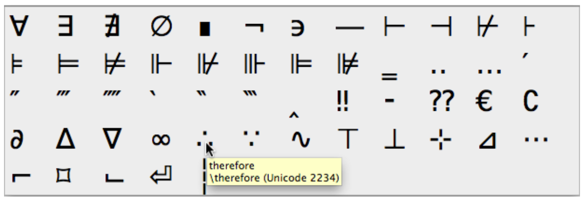
Practice Exercises

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



(1) It is raining heavily.
(2) If you do not take an umbrella, you will get soaked.
∴ You should take an umbrella.

Figure 2.1: The argument from page 12 entered into *Lurch*. (The font has been enlarged here to show details more easily, a convention I will follow throughout the text.)



therefore
\therefore (Unicode 2234)

Figure 2.2: Hovering the mouse over the ∴ symbol to learn its details

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. 17, 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

Chapter 3

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.

3.1 Sentence letters

In SL, capital letters 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:

There is an apple on the desk.
If there is an apple on the desk, then Jenny made it to class.
∴ Jenny made it to class.

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: There is an apple on the desk.
B: If there is an apple on the desk, then Jenny made it to class.
C: Jenny made it to class.

We would then symbolize the argument in this way:

A
B
∴ C

But this is a terrible way to symbolize this argument, because 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. The structure of the argument has been completely lost in this translation.

The important thing about the argument is that its second premise is not merely *any* sentence, logically divorced from the other sentences in the argument. 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.
$\therefore$ C

This preserves the structure of the argument that makes it valid, but it still makes use of the English expression 'If… then….' Although 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.

There are only twenty-six letters of the alphabet, but there is no logical limit to the number of atomic sentences. Therefore we will permit atomic sentences to be comprised of one *or more* letters. We could have a symbolization key that looks like this:

Ap:	The apple is under the armoire.
Ar:	Arguments in SL always contain atomic sentences.
Ad:	Adam Ant is taking an airplane from Anchorage to Albany.
All:	Alliteration angers otherwise affable astronauts.
⋮	(and so on)

Keep in mind that each of these is a different atomic sentence.

3.2 Connectives

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

symbol	what it is called	what it means
$\neg$	negation	'It is not the case that…'
$\wedge$	conjunction	'Both… and …'
$\vee$	disjunction	'Either… or …'
$\Rightarrow$	conditional	'If … then …'
$\Leftrightarrow$	biconditional	'… if and only if …'

Negation

Consider how we might symbolize these sentences:

1. Mary is in Barcelona.
2. Mary is not in Barcelona.

3. Mary is somewhere besides Barcelona.

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

B: Mary is in Barcelona.

Note that here we are giving B a different interpretation than we did in the previous section. The symbolization key only specifies what B means *in a specific context.* It is vital that we continue to use this meaning of B so long as we are talking about Mary and Barcelona. Later, when we are symbolizing different sentences, we can write a new symbolization key and use B to mean something else.

Now, sentence 1 is simply B.

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

Sentence 3 is about whether or not Mary is in Barcelona, but it does not contain the word 'not.' Nevertheless, it is obviously logically equivalent to sentence 2. They both mean: It is not the case that Mary is in Barcelona. As such, we can translate both sentence 2 and sentence 3 as $\neg B$.

A sentence can be symbolized as $\neg\mathscr{A}$ if it can be paraphrased in English as 'It is not the case that $\mathscr{A}$.'

Consider these further examples:

4. The widget can be replaced if it breaks.
5. The widget is irreplaceable.
6. The widget is not irreplaceable.

If we let R mean 'The widget is replaceable', then sentence 4 can be translated as R.

What about sentence 5? Saying the widget is irreplaceable means that it is not the case that the widget is replaceable. So even though sentence 5 is not negative in English, we symoblize it using negation as $\neg R$.

Sentence 6 can be paraphrased as 'It is not the case that the widget is irreplaceable.' 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.' If you think about the sentence in English, it is logically equivalent to sentence 4. So when we define logical equivalence in SL, we will make sure that R and $\neg\neg R$ are logically equivalent.

More examples:

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

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

However, it would be a mistake to symbolize sentence 8 as $\neg H$. If Elliott is unhappy, then he is not happy—but sentence 8 does not mean the same thing as 'It is not the case that Elliott is happy.' It could be that he is not happy but that he is not unhappy either. Perhaps he is somewhere between the two. In order to symbolize sentence 8, we would need a new sentence letter.

For any sentence $\mathcal{A}$: If $\mathcal{A}$ is true, then $\neg\mathcal{A}$ is false. If $\neg\mathcal{A}$ is true, then $\mathcal{A}$ is false. Using 'T' for true and 'F' for false, we can summarize this in a *characteristic truth table* for negation:

$\mathcal{A}$	$\neg\mathcal{A}$
T	F
F	T

We will discuss truth tables at greater length in the next chapter.

Conjunction

Consider these sentences:

9. Adam is athletic.
10. Barbara is athletic.
11. Adam is athletic, and Barbara is also athletic.

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

A: Adam is athletic.
B: Barbara is athletic.

Sentence 9 can be symbolized as A.

Sentence 10 can be symbolized as B.

Sentence 11 can be paraphrased as 'A and B.' In order to fully symbolize this sentence, we need another symbol. We will use '$\wedge$.' We translate 'A and B' as $A \wedge B$. The logical connective '$\wedge$' is called CONJUNCTION, and A and B 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.

Some more examples:

12. Barbara is athletic and energetic.
13. Barbara and Adam are both athletic.
14. Although Barbara is energetic, she is not athletic.
15. Barbara is athletic, but Adam is more athletic than she is.

Sentence 12 is obviously a conjunction. The sentence says two things about Barbara, so in English it is permissible to use to Barbara's name only once. It might be tempting to try this when translating the argument: Since B means 'Barbara is athletic', one might paraphrase the sentences as 'B and energetic.' This would be a mistake. Once we translate part of a sentence as B, any further structure is lost. B is an atomic sentence; it is nothing more than true or false. Conversely, 'energetic' is not a sentence; on its own it is neither true nor false. We should instead paraphrase the sentence as 'B and Barbara is energetic.' Now we need to add a sentence letter to the symbolization key. Let E mean 'Barbara is energetic.' Now the sentence can be translated as $B \wedge E$.

A sentence can be symbolized as $\mathcal{A} \wedge \mathcal{B}$ if it can be paraphrased in English as 'Both $\mathcal{A}$, and $\mathcal{B}$.' Each of the conjuncts must be a sentence.

Sentence 13 says one thing about two different subjects. It says of both Barbara and Adam that they are athletic, and in English we use the word 'athletic' only once. In translating to SL, it is important to realize that the sentence can be paraphrased as, 'Barbara is athletic, and Adam is athletic.' This translates as $B \wedge A$.

Sentence 14 is a bit more complicated. The word 'although' sets up a contrast between the first part of the sentence and the second part. Nevertheless, the sentence says both that Barbara is energetic and that she is not athletic. In order to make each of the conjuncts an atomic sentence, we need to replace 'she' with 'Barbara.'

So we can paraphrase sentence 14 as, '*Both* Barbara is energetic, *and* Barbara is not athletic.' The second conjunct contains a negation, so we paraphrase further: '*Both* Barbara is energetic *and it is not the case that* Barbara is athletic.' This translates as $E \wedge \neg B$.

Sentence 15 contains a similar contrastive structure. It is irrelevant for the purpose of translating to SL, so we can paraphrase the sentence as '*Both* Barbara is athletic, *and* Adam is more athletic than Barbara.' (Notice that we once again replace the pronoun 'she' with her name.) How should we translate the second conjunct? We already have the sentence letter A which is about Adam's being athletic and B which is about Barbara's being athletic, but neither is about one of them being more athletic than the other. We need a new sentence letter. Let R mean 'Adam is more athletic than Barbara.' Now the sentence translates as $B \wedge R$.

Sentences that can be paraphrased '$\mathcal{A}$, but $\mathcal{B}$' or 'Although $\mathcal{A}$, $\mathcal{B}$' are best symbolized using conjunction: $\mathcal{A} \wedge \mathcal{B}$

It is important to keep in mind that the sentence letters A, B, and R are atomic sentences. Considered as symbols of SL, they have no meaning beyond being true or false. We have used them to symbolize different English language sentences that are all about people being athletic, but this similarity is completely lost when we translate to SL. No formal language can capture all the structure of the English language, but as long as this structure is not important to the argument there is nothing lost by leaving it out.

For any sentences $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A} \wedge \mathcal{B}$ is true if and only if both $\mathcal{A}$ and $\mathcal{B}$ are true. We can summarize this in the characteristic truth table for conjunction:

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A} \wedge \mathcal{B}$
T	T	T
T	F	F
F	T	F
F	F	F

Conjunction is *symmetrical* because we can swap the conjuncts without changing the truth value of the sentence. Regardless of what $\mathcal{A}$ and $\mathcal{B}$ are, $\mathcal{A} \wedge \mathcal{B}$ is logically equivalent to $\mathcal{B} \wedge \mathcal{A}$.

Quiz Yourself

- If A stands for "Apples taste good," what does $\neg A$ mean?
- Are A and $\neg A$ mutually exclusive?
- Explain the meaning of each line in the characteristic truth table for conjunction.

Disjunction

Consider these sentences:

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

For these sentences we can use this symbolization key:

D:	Denison will play golf with me.
E:	Ellery will play 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 $D \vee M$. The '$\vee$' 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 uses 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 $D \vee E$.

A sentence can be symbolized as $\mathcal{A} \vee \mathcal{B}$ if it can be paraphrased in English as 'Either $\mathcal{A}$, or $\mathcal{B}$.' Each of the disjuncts must be a sentence.

Sometimes in English, the word 'or' excludes the possibility that both disjuncts are true. This is called an EXCLUSIVE OR. An *exclusive or* is clearly intended when it says, on a restaurant menu, 'Entrees come with either soup or salad.' You may have soup; you may have salad; but, if you want *both* soup *and* salad, then you have to pay extra.

At other times, the word 'or' allows for the possibility that both disjuncts might be true. This is probably the case with sentence 17, above. I might play with Denison, with Ellery, or with both Denison and Ellery. Sentence 17 merely says that I will play with *at least* one of them. This is called an INCLUSIVE OR.

The symbol '$\vee$' represents an *inclusive or*. So $D \vee E$ 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:

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A} \vee \mathcal{B}$
T	T	T
T	F	T
F	T	T
F	F	F

Like conjunction, disjunction is symmetrical. $\mathscr{A} \vee \mathscr{B}$ is logically equivalent to $\mathscr{B} \vee \mathscr{A}$.

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 either soup or salad, but not both.

We let So mean that you get soup and Sa mean that you 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 $\neg So \vee \neg Sa$.

Sentence 19 also requires negation. It can be paraphrased as, '*It is not the case that* either you get soup or 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, just like you would do to group terms in mathematics: 'It is not the case that $(So \vee Sa)$.' This becomes simply $\neg(So \vee Sa)$.

Notice that the parentheses are doing important work here. If we left them out, we would have the sentence $\neg So \vee Sa$, which means 'Either you will not have soup, or you will have salad.' That is not our intended meaning.

Sentence 20 is an *exclusive or*. We can break the sentence into two parts. The first part says that you get one or the other. We translate this as $(So \vee Sa)$. 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 $\neg(So \wedge Sa)$. 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 $(So \vee Sa) \wedge \neg(So \wedge Sa)$.

Although '$\vee$' is an *inclusive or*, we can symbolize an *exclusive or* in SL. We just need more than one connective to do it.

Conditional

For the following sentences, let R mean 'You will cut the red wire' and B mean '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 '$\Rightarrow$' to represent logical entailment, the idea that one statement leads us to another. The sentence becomes $R \Rightarrow B$. 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 $R \Rightarrow B$ 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 $R \Rightarrow B$ 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 involve your having 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 $B \Rightarrow R$.

It is important to remember that the connective '$\Rightarrow$' 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 $B \Rightarrow R$ 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 '$\mathcal{A}$ only if $\mathcal{B}$' is logically equivalent to 'If $\mathcal{A}$, then $\mathcal{B}$.'

'If $\mathcal{A}$ then $\mathcal{B}$' means that if $\mathcal{A}$ is true then so is $\mathcal{B}$. So we know that if the antecedent $\mathcal{A}$ is true but the consequent $\mathcal{B}$ is false, then the conditional 'If $\mathcal{A}$ then $\mathcal{B}$' is false. What is the truth value of 'If $\mathcal{A}$ then $\mathcal{B}$' under other circumstances? Suppose, for instance, that the antecedent $\mathcal{A}$ happened to be false. 'If $\mathcal{A}$ then $\mathcal{B}$' would then not tell us anything about the actual truth value of the consequent $\mathcal{B}$, and it is unclear what the truth value of 'If $\mathcal{A}$ then $\mathcal{B}$' 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 problem 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 the English language 'If... then....' Instead, the symbol '$\Rightarrow$' will be a *material conditional.* This means that when $\mathcal{A}$ is false, the conditional $\mathcal{A} \Rightarrow \mathcal{B}$ is automatically true, regardless of the truth value of $\mathcal{B}$. If both $\mathcal{A}$ and $\mathcal{B}$ are true, then the conditional $\mathcal{A} \Rightarrow \mathcal{B}$ is true.

In short, $\mathcal{A} \Rightarrow \mathcal{B}$ is false if and only if $\mathcal{A}$ is true and $\mathcal{B}$ is false. We can summarize this with a characteristic truth table for the conditional.

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A} \Rightarrow \mathcal{B}$
T	T	T
T	F	F
F	T	T
F	F	T

The conditional is our only *asymmetrical* symbol. You cannot swap the antecedent and consequent without changing the meaning of the sentence, because $\mathcal{A}\Rightarrow\mathcal{B}$ and $\mathcal{B}\Rightarrow\mathcal{A}$ are not logically equivalent.

Not all sentences of the form 'If... then...' are conditionals. Consider this sentence:

23. If anyone wants to see me, then I will be on the porch.

If I say this, it means that I will be on the porch, regardless of whether anyone wants to see me or not— but if someone did want to see me, then they should look for me there. If we let P mean 'I will be on the porch,' then sentence 23 can be translated simply as P.

Biconditional

Consider these sentences:

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

Let T mean 'The figure is a triangle' and S mean 'The figure has three sides.'

Sentence 24, for reasons discussed above, can be translated as $T \Rightarrow S$.

Sentence 25 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 $S \Rightarrow T$.

Sentence 26 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 $S \Rightarrow T$ and $T \Rightarrow S$. We will use '$\Leftrightarrow$' to represent the biconditional; sentence 26 can be translated as $S \Leftrightarrow T$.

We could abide without a new symbol for the biconditional. Since sentence 26 means '$T \Rightarrow S$ and $S \Rightarrow T$,' we could translate it as $(T \Rightarrow S) \wedge (S \Rightarrow T)$. We would need parentheses to indicate that $(T \Rightarrow S)$ and $(S \Rightarrow T)$ are separate conjuncts; the expression $T \Rightarrow S \wedge S \Rightarrow T$ would be ambiguous.

Because we could always write $(\mathcal{A} \Rightarrow \mathcal{B}) \wedge (\mathcal{B} \Rightarrow \mathcal{A})$ instead of $\mathcal{A} \Leftrightarrow \mathcal{B}$, 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.'

$\mathcal{A} \Leftrightarrow \mathcal{B}$ is true if and only if $\mathcal{A}$ and $\mathcal{B}$ have the same truth value. Thus the biconditional is also symmetrical, and this is its characteristic truth table:

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A} \Leftrightarrow \mathcal{B}$
T	T	T
T	F	F
F	T	F
F	F	T

3.3 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':

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

Let J mean 'You will wear a jacket' and let D mean 'You will catch a cold.'

We can paraphrase sentence 27 as 'Unless J, D.' This means that if you do not wear a jacket, then you will catch cold; with this in mind, we might translate it as $\neg J \Rightarrow D$. 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 $\neg D \Rightarrow J$.

Which of these is the correct translation of sentence 27? Both translations are correct, because the two translations are logically equivalent in SL.

Sentence 28, in English, is logically equivalent to sentence 27. It can be translated as either $\neg J \Rightarrow D$ or $\neg D \Rightarrow J$.

When symbolizing sentences like sentence 27 and sentence 28, it is easy to get turned around. Since the conditional is not symmetric, it would be wrong to translate either sentence as $J \Rightarrow \neg D$. 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 $J \vee D$. (You might worry that the 'or' here should be an *exclusive or.* However, the sentences do not exclude the possibility that you might *both* wear a jacket *and* catch a cold; jackets do not protect you from all the possible ways that you might catch a cold.)

If a sentence can be paraphrased as 'Unless $\mathcal{A}$, $\mathcal{B}$,' then it can be symbolized as $\mathcal{A} \vee \mathcal{B}$.

Symbolization of standard sentence types is summarized on p. 199.

Quiz Yourself

- Which line of the truth table for the conditional is the least intuitive? Why?
- Give at least five English sentence forms that we now know how to translate into SL.
- Which of the connectives introduced so far did the text say is unnecessary?

3.4 Sentences of SL

The sentence 'Apples are red, or berries are blue' is a sentence of English, and the sentence '$(A \vee B)$' 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. The sentence '$(A \vee B)$' 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.

Expressions

There are three kinds of symbols in SL:

sentence letters or multi-letter names	$A, B, C, \ldots, Z$ $AA, AB, AC, \ldots, ZA, \ldots, Tina, \ldots$
connectives	$\neg$,$\wedge$,$\vee$,$\Rightarrow$,$\Leftrightarrow$
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.

Well-formed formulae

Since any sequence of symbols is an expression, many expressions of SL will be gobbledegook. A meaningful expression is called a *well-formed formula.* It is common to use the acronym *wff*, pronounced the same as 'woof.' The plural is wffs.

Obviously, individual atomic sentences like A and $Gary$ will be wffs. We can form further wffs out of these by using the various connectives. Using negation, we can get $\neg A$ and $\neg Gary$. Using conjunction, we can get $A \wedge Gary$, $Gary \wedge A$, $A \wedge A$, and $Gary \wedge Gary$. We could also apply negation repeatedly to get wffs like $\neg\neg A$ or apply negation along with conjunction to get wffs like $\neg(A \wedge Gary)$ and $\neg(Gary \wedge \neg Gary)$. 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 process by which wffs can be constructed. Consider negation: Given any wff $\mathcal{A}$ of SL, $\neg\mathcal{A}$ is a wff of SL. It is important here that $\mathcal{A}$ is not the sentence letter A. Rather, it is a variable that stands in for any wff at all. Notice that this variable $\mathcal{A}$ is not a symbol of SL, so $\neg\mathcal{A}$ 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 $\mathcal{A}$ is part of the metalanguage, it is called a *metavariable*.

We can say similar things for each of the other connectives. For instance, if $\mathcal{A}$ and $\mathcal{B}$ are wffs of SL, then $(\mathcal{A} \wedge \mathcal{B})$ is a wff of SL. 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. If $\mathcal{A}$ is a wff, then $\neg\mathcal{A}$ is a wff of SL.
3. If $\mathcal{A}$ and $\mathcal{B}$ are wffs, then $(\mathcal{A} \wedge \mathcal{B})$ is a wff.
4. If $\mathcal{A}$ and $\mathcal{B}$ are wffs, then $(\mathcal{A} \vee \mathcal{B})$ is a wff.
5. If $\mathcal{A}$ and $\mathcal{B}$ are wffs, then $(\mathcal{A} \Rightarrow \mathcal{B})$ is a wff.
6. If $\mathcal{A}$ and $\mathcal{B}$ are wffs, then $(\mathcal{A} \Leftrightarrow \mathcal{B})$ is a wff.
7. All and only wffs of SL can be generated by applications of these rules.

Notice that we cannot immediately apply this definition to see whether an arbitrary expression is a wff. 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.

Definitions like this are called *recursive.* Recursive definitions begin with some specifiable base elements and define ways to indefinitely compound the base elements. Just as the recursive definition allows complex sentences to be built up from simple parts, you can use it to decompose sentences into their simpler parts. To determine whether or not something meets the definition, you may have to refer back to the definition many times.

The connective that you look to first in decomposing a sentence is called the MAIN LOGICAL OPERATOR of that sentence. For example: The main logical operator of $\neg(E \vee (F \Rightarrow G))$ is negation, $\neg$. The main logical operator of $(\neg E \vee (F \Rightarrow G))$ is disjunction, $\vee$.

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 the next chapter.

Notational conventions

A wff like $(Q \wedge R)$ 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 $(Q \wedge R)$, we get $\neg(Q \wedge R)$. If we just had $Q \wedge R$ without the parentheses and put a negation in front of it, we would have $\neg Q \wedge R$. It is most natural to read this as meaning the same thing as $(\neg Q \wedge R)$, something very different than $\neg(Q \wedge R)$. The sentence $\neg(Q \wedge R)$ 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 $(\neg Q \wedge R)$ means specifically that Q is false and that R is true. As such, parentheses are crucial to the meaning of the sentence.

So, strictly speaking, $Q \wedge R$ without parentheses is *not* a sentence of SL. When using SL, however, we will often 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 $Q \wedge R$ means the same thing as $(Q \wedge R)$. As a matter of convention, we can leave off parentheses that occur *around the entire sentence.*

Second, 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 $(A \wedge B) \wedge C$ or as $A \wedge (B \wedge C)$. 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 $(A \wedge B)$ is conjoined with C, and the second, in which A is conjoined with $(B \wedge C)$. So we might as well just write $A \wedge B \wedge C$. As a matter of convention, we can leave out parentheses when we conjoin three or more sentences.

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

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 $(A \wedge B) \vee C$ and $A \wedge (B \vee C)$. The parentheses are also required if there is a series of conditionals or biconditionals; as with $(A \Rightarrow B) \Rightarrow C$ and $A \Leftrightarrow (B \Leftrightarrow C)$.

We have adopted these three rules as *notational conventions*, not as changes to the definition of a sentence. Strictly speaking, $A \vee B \vee C$ 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 $((A \vee B) \vee C)$.

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 $\mathcal{A}$, $\mathcal{B}$, …, $\mathcal{Z}$ are wffs, then $(\mathcal{A} \wedge \mathcal{B} \wedge \ldots \wedge \mathcal{Z})$, 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. Adopting notational conventions is a compromise between these two desires.

3.5 How *Lurch* can help

In Chapter 2, I encouraged you to type your homework in *Lurch*, but there was minimal benefit to your doing so, other than knowing where to find the $\therefore$symbol. In this chapter, however, *Lurch* can give you feedback on your work as you do it.

Several of this chapter's practice problems are about wffs, a concept *Lurch* knows. To leverage *Lurch*'s knowledge in this area, we need to know how to type in the symbols from this chapter, and how to get *Lurch* to pay attention to them.

Entering wff symbols into *Lurch*

The five logical connectives we've seen so far can be used in *Lurch* as follows.

connective	found on this symbol menu	keyboard shortcut(s) (choose your favorite)
$\neg$	$\forall$	`\neg`
$\wedge$	$\pm$	`\and` `\wedge`
$\vee$	$\pm$	`\or` `\vee`
$\Rightarrow$	$\rightarrow$	`\implies` `\then` `\Rightarrow`
$\Leftrightarrow$	$\rightarrow$	`\iff`

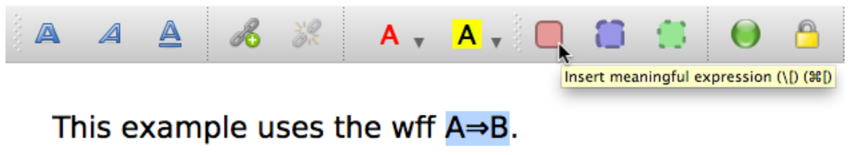
The shortcut `\iff` is so named because 'iff' is a common mathematical shorthand for 'if and only if.'

So, for example, you could type the following sequence of keystrokes in *Lurch*.

```
(A \and B) \then (C \or D)
```

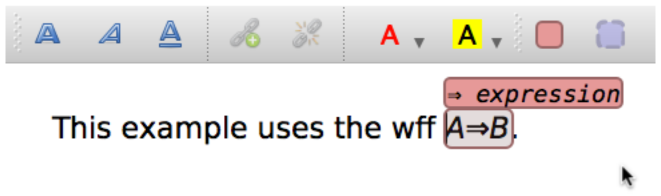
And *Lurch* would convert your text, as you type it, into the following form.

$$(A \wedge B) \Rightarrow (C \vee D)$$



This example uses the wff A⇒B.

Figure 3.1: A highlighted section of text about to become a meaningful expression



This example uses the wff A⇒B.

Figure 3.2: The text from Figure 3.1 now marked as meaningful

Having *Lurch* pay attention

Lurch does not try to read or understand anything you type unless you specifically tell it to do so. By default, it is a plain word processor that doesn't use its mathematical abilities. To get it to pay attention to the meaning of the symbols you're typing, you use the 'Meaning' menu (and its corresponding toolbar).

To use it, first type a wff into *Lurch* and then highlight it with your cursor, just as if you were about to make it bold or italic, as shown in Figure 3.1. Then click the red button under the mouse cursor in that figure, and you will find your wff wrapped in a red bubble and labeled, as in Figure 3.2.

The red bubble is *Lurch*'s way of telling you that it is paying attention to the meaning of that section of text. Any unbubbled text is ignored by the software. When your cursor moves outside of the bubble, *Lurch* will hide the bubble, so as not to clutter up your document. When your cursor re-enters that text, the bubble reappears.

The tag on the top of the bubble, '⇒ expression,' indicates two things. First, it says that *Lurch* understands what the contents of the bubble mean. If it did not understand them, it would have said 'string' in that tag, to mean, 'This is a string of symbols but I can't see a meaning to it.' Second, it's telling us that the expression's main logical operator is the ⇒. That is, this wff is a ⇒ applied to two smaller wffs.

To delete a bubble you accidentally inserted, use the 'Remove bubble' action from the meaning menu or toolbar, or place your cursor immediately inside the left edge of the bubble and backspace over the bubble's boundary.

What good is that?

If you use *Lurch* when writing solutions to the following practice problems, you'll therefore derive three benefits.

1. Easy access to all the symbols you need, either on menus or through keyboard shortcuts, as you prefer.
2. Immediate feedback from *Lurch* on whether a sequence of symbols is a wff or not. (Note that *Lurch* permits all the notational conventions listed earlier in this chapter.)

3. Immediate feedback from *Lurch* on which logical operator in a wff is the main one.

This will not help you answer every single question in the pracice problems, but it can be quite handy. *Lurch* has a lot more power than this; we're only beginning to tap its potential. We'll see more in future chapters.

Quiz Yourself

- In this text, is SL the object language or the metalanguage?
- What are the smallest wffs?
- What is the significance of the final rule for forming wffs?
- If you entered the wff $A \Rightarrow B$ into a meaningful expression in *Lurch*, what would it show on the tag above the expression?

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.

AE:	Ava is an electrician.
HE:	Harrison is an electrician.
AF:	Ava is a firefighter.
HF:	Harrison is a firefighter.
AS:	Ava is satisfied with her career.
HS:	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 What text does *Lurch* put on the tag of a bubble containing an atomic wff?

⋆ **Part E** 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 F 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 G 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 H** For each of the following: (a) Is it a wff of SL? (b) Is it a sentence of SL, allowing for notational conventions?

1. (A)
2. $Jay \vee \neg Jay$
3. $\neg\neg\neg\neg F$
4. $\neg \wedge S$
5. $(G \wedge \neg G)$
6. $\mathcal{A} \Rightarrow \mathcal{A}$
7. $(A \Rightarrow (A \wedge \neg F)) \vee (D \Leftrightarrow E)$
8. $((Z \Leftrightarrow S) \Rightarrow W) \wedge (J \vee X)$
9. $(F \Leftrightarrow \neg D \Rightarrow J) \vee (C \wedge D)$

Part I

1. Are there any wffs of SL that contain no sentence letters? Why or why not?
2. In the chapter, we symbolized an *exclusive or* using $\vee$, $\wedge$, and $\neg$. How could you translate an *exclusive or* using only two connectives? Is there any way to translate an *exclusive or* using only one connective?

Chapter 4

Truth tables

This chapter introduces a way of evaluating sentences and arguments of SL. Although it can be laborious, the truth table method is a purely mechanical procedure that requires no intuition or special insight.

4.1 Truth-functional connectives

Any non-atomic sentence of SL is composed of atomic sentences and logical connectives. The truth value of the compound sentence depends only on the truth value of the atomic sentences that comprise it. In order to know the truth value of $(D \Leftrightarrow E)$, for instance, you only need to know the truth value of D and the truth value of E. Connectives that work in this way are called TRUTH-FUNCTIONAL.

In this chapter, we will make use of the fact that all of the logical operators in SL are truth-functional— it makes it possible to construct truth tables to determine the logical features of sentences. You should realize, however, that this is not possible for all languages. In English, you can take any sentence $\mathcal{X}$ and form a new sentence from it by saying 'It is possible that $\mathcal{X}$.' The truth value of this new sentence does not depend directly on the truth value of $\mathcal{X}$. Even if $\mathcal{X}$ is false, perhaps in some sense $\mathcal{X}$ *could* have been true— then the new sentence would be true. Some formal languages, called *modal logics*, have an operator for possibility. In a modal logic, we could translate 'It is possible that $\mathcal{X}$' as $\Diamond\mathcal{X}$. However, the ability to translate sentences like these come at a cost: The $\Diamond$ operator is not truth-functional, and so modal logics are not amenable to truth tables.

4.2 Complete truth tables

The truth value of sentences that contain only one connective is given by the characteristic truth table for that connective. To put them all in one place, the truth tables for the connectives of SL are repeated in table 4.1.

The characteristic truth table for conjunction, for example, gives the truth conditions for any sentence of the form $(\mathcal{A} \wedge \mathcal{B})$. Even if the conjuncts $\mathcal{A}$ and $\mathcal{B}$ are long, complicated sentences, the conjunction is true if and only if both $\mathcal{A}$ and $\mathcal{B}$ are true. Consider the sentence $(H \wedge I) \Rightarrow H$. 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.

$\mathcal{A}$	$\neg\mathcal{A}$
T	F
F	T

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A}\wedge\mathcal{B}$	$\mathcal{A}\vee\mathcal{B}$	$\mathcal{A}\Rightarrow\mathcal{B}$	$\mathcal{A}\Leftrightarrow\mathcal{B}$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

Table 4.1: The characteristic truth tables for the connectives of SL.

H	I	$(H \wedge I) \Rightarrow H$
T	T	**T** **T** **T**
T	F	**T** **F** **T**
F	T	**F** **T** **F**
F	F	**F** **F** **F**

Now consider the subsentence $H \wedge I$. This is a conjunction $\mathcal{A} \wedge \mathcal{B}$ with H as $\mathcal{A}$ and with I as $\mathcal{B}$. H and I are both true on the first row. Since a conjunction is true when both conjuncts are true, we write a T underneath the conjunction symbol. We continue for the other three rows and get this:

H	I	$(H \wedge I) \Rightarrow H$
		$\mathcal{A}$ $\wedge$ $\mathcal{B}$
T	T	T **T** T T
T	F	T **F** F T
F	T	F **F** T F
F	F	F **F** F F

The entire sentence is a conditional $\mathcal{A} \Rightarrow \mathcal{B}$ with $(H \wedge I)$ as $\mathcal{A}$ and with H as $\mathcal{B}$. On the second row, for example, $(H \wedge I)$ is false and H is true. Since a conditional is true when the antecedent is false, we write a T in the second row underneath the conditional symbol. We continue for the other three rows and get this:

H	I	$(H \wedge I) \Rightarrow H$
		$\mathcal{A}$ $\Rightarrow$ $\mathcal{B}$
T	T	T **T** T
T	F	F **T** T
F	T	F **T** F
F	F	F **T** F

The column of Ts underneath the conditional tells us that the sentence $(H \wedge I) \Rightarrow H$ 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 combination of truth values we did not investigate.

In this example, we have not repeated all of the entries in every successive table. 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	$(H \wedge I) \Rightarrow H$
T	T	T T T T T
T	F	T F F T T
F	T	F F T T F
F	F	F F F T F

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.

A COMPLETE TRUTH TABLE has a row for all the possible combinations of T and F 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 $((C \Leftrightarrow C) \Rightarrow C) \wedge \neg(C \Rightarrow C)$. 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 T and F on the same row. The truth table for this sentence looks like this:

C	$((C$	$\Leftrightarrow$	$C)$	$\Rightarrow$	$C)$	$\wedge$	$\neg$	$(C$	$\Rightarrow$	$C)$
T	T	T	T	T	T	**F**	F	T	T	T
F	F	T	F	F	F	**F**	F	F	T	F

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.

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

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

M	N	P	M	$\wedge$	$(N$	$\vee$	$P)$
T	T	T	T	**T**	T	T	T
T	T	F	T	**T**	T	T	F
T	F	T	T	**T**	F	T	T
T	F	F	T	**F**	F	F	F
F	T	T	F	**F**	T	T	T
F	T	F	F	**F**	T	T	F
F	F	T	F	**F**	F	T	T
F	F	F	F	**F**	F	F	F

From this table, we know that the sentence $M \wedge (N \vee P)$ 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. Five letters, 32 lines. Six letters, 64 lines. And so on. To be perfectly 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 Ts and Fs. In the next column to the left, write two Ts, write two Fs, and repeat. For the third sentence letter, write four Ts followed by four Fs. 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 Ts followed by eight Fs. For a 32 line table, the next column would have 16 Ts followed by 16 Fs. And so on.

Quiz Yourself

- What does it mean for a truth table to be complete?
- When building a truth table, do you begin with the sentence letters or with the main logical connective?
- Name a few positive whole numbers that cannot equal the number of lines in any complete truth table.

4.3 Using truth tables

Tautologies, contradictions, and contingent sentences

Recall 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 T on every row of a complete truth table.

Conversely, a sentence is a CONTRADICTION IN SL if the column under its main connective is F 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 T on at least one row and F on at least one row.

From the truth tables in the previous section, we know that $(H \wedge I) \Rightarrow H$ is a tautology, that $((C \Leftrightarrow C) \Rightarrow C) \wedge \neg(C \Rightarrow C)$ is a contradiction, and that $M \wedge (N \vee P)$ is contingent.

Logical equivalence

Two sentences are logically equivalent in English if they have the same truth value as a matter 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 $\neg A \wedge \neg B$. Are they logically equivalent? To find out, we construct a truth table.

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

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

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.

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 T and the conclusion is F; an argument is INVALID IN SL if there is such a row.

Consider this argument:

$$\begin{array}{ll} & \neg L \Rightarrow (J \vee L) \\ & \neg L \\ \therefore & J \end{array}$$

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

J	L	$\neg\, L \Rightarrow (J \vee L)$	¬ L	J
T	T	F T **T** T T T	**F** T	**T**
T	F	T F **T** T T F	**T** F	**T**
F	T	F T **T** F T T	**F** T	**F**
F	F	T F **F** F F F	**T** F	**F**

Yes, the argument is valid. The only row on which both the premises are T is the second row, and on that row the conclusion is also T.

4.4 Partial truth tables

In order to show that a sentence is a tautology, we need to show that it is T 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 F. 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 $(U \wedge T) \Rightarrow (S \wedge W)$. We want to show that it is *not* a tautology by providing a partial truth table. We fill in F 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 (T) and the consequent must be false (F). So we fill these in on the table:

S	T	U	W	$(U \wedge T) \Rightarrow (S \wedge W)$
				T **F** F

In order for the $(U \wedge T)$ to be true, both U and T must be true.

S	T	U	W	$(U \wedge T) \Rightarrow (S \wedge W)$
	T	T		T T T **F** F

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

Now we just need to make $(S \wedge W)$ 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	$(U \wedge T) \Rightarrow (S \wedge W)$
F	T	T	F	T T T **F** F F F

Showing that something is a contradiction requires a complete truth table. Showing that something is *not* a contradiction requires only a one-line partial truth table, where the sentence is true on that one line.

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	$(U \wedge T) \Rightarrow (S \wedge W)$
F	T	T	F	T T T **F** F F F
F	T	F	F	F F T **T** F F F

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.

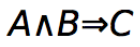
Showing that a sentence is *not* contingent requires providing a complete truth table, because it requires showing that the sentence is a tautology or that it is a contradiction. If you do not know whether a particular sentence is contingent, then you do not know whether you will need a complete or partial 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 4.2 summarizes when a complete truth table is required and when a partial truth table will do.



$A \wedge B \Rightarrow C$

Figure 4.1: A meaningful expression with no visible bubble, because the cursor is elsewhere

4.5 *Lurch* and truth tables

Recall what the first paragraph of this chapter introduced about truth tables: They can take awhile to create, but doing so is like long division, in that you don't need to be creative or have great ideas; you just learn the rules and do it.

Lurch is targeted at more complex areas of logic that we'll begin next chapter, in which you *do* need to be creative and come up with insightful ideas. Consequently, it doesn't know or care about truth tables, and so it can only provide you a small amount of help with this chapter's practice problems.

Of course, you can continue to type your homework in *Lurch*, lining up columns of truth tables with the Tab key, and entering the logical symbols as you did before. And there is one, small, additional benefit *Lurch* can provide as set up a truth table. But it can't do the truth table for you, nor can it check one that you have done.

Sentence structure

You saw at the start of Section 4.2 how each row in a truth table is formed in a specific order. We start by knowing the truth values for the atomic sentences, and then work as we do in algebra, evaluating parentheses first, and working from the inside out. The final step assigns a truth value to the main logical connective of the sentence.

Or, if you're working backwards to creat a partial truth table, as in Section 4.4, you might go in exactly the reverse order. Beginning with the truth value you want the main connective to have, you work downwards through the sentence's connectives until you reach the atomic sentences.

Both of these require you to understand the structure of a wff. Although the notation of SL encourages using parentheses to make this structure of wffs clear, *Lurch* can help if you're having trouble seeing it.

Expanded form

In the previous chapter, I mentioned that the red bubbles for meaningful expressions in *Lurch* only appear when your cursor is inside them. So, for example, the meaningful expression shown in Figure 4.1 does not show its red bubble because the cursor is elsewhere.

At times, however, you may wish to know where all the bubbles in your document are, even without moving your cursor around to find them. For this reason, *Lurch* enables you to toggle on and off the display of bubble boundaries as colored brackets. On the Meaning menu, the command is called 'Show markers around bubbles,' and its default keyboard shortcut is Control-1 (or Command-1 on a Mac). It would show the expression from Figure 4.1 with red brackets for bubble boundaries, even when the cursor is not inside the expression, as in Figure 4.2.

To see the structure of the expression, right-click anywhere inside the bubble (control-click on a Mac) and choose 'Expand' from the context menu. The result will look like Figure 4.3. Wow! Let's consider what that structure means.

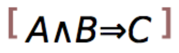

[A∧B⇒C]

Figure 4.2: A *Lurch* meaningful expression with bubble boundaries diplayed as brackets

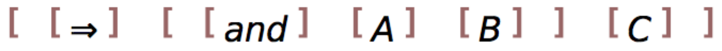

[[⇒] [[and] [A] [B]] [C]]

Figure 4.3: An expanded meaningful expression

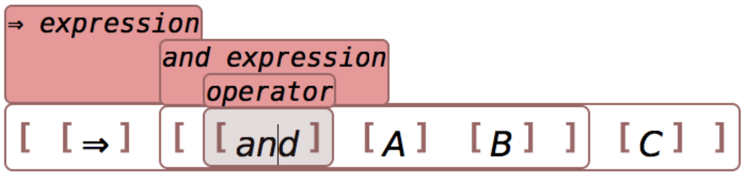
First, the red brackets work like parentheses; they group things together. So the outermost brackets make the entire meaningful expression one large group. Within that larger group, we can see three smaller groups, the first containing only the symbol $\Rightarrow$, the second containing several things, and the last one containing only the atomic sentence C.

Second, *Lurch*'s expanded form always puts the main logical connective in each group *first*. So the $\Rightarrow$ symbol appears first in the outermost group because it is the main connective in the expression. Within the middle group, the word 'and' appears first, because it is the main connective in that inner expression. Thus expanded form shows the hierarchical structure inherent in the logical expression, whether or not it was originally written with parentheses for clarity.

If you place your cursor in one of the inner bubbles, you will see that *Lurch* shows you information about all of the bubbles your cursor is inside, not just the innermost one. See Figure 4.4. *Lurch* uses the word 'operator' to mean what our text has been calling a 'logical connective,' because *Lurch* can handle many kinds of operators, not just ones relating to logic.

To return your expression to the form it was originally in, put your cursor somewhere in the outermost bubble (but not inside an inner bubble) and right-click it (control-click on a Mac). Choose 'Collapse' to reverse the earlier expanding process. The bubble context menu also responds to the keyboard shortcut Control-Enter (Command-Return on a Mac).

So *Lurch* cannot help you form truth tables, but it can show you the hierarchical structure of your logical expressions, which you need to use when building a truth table. I suggest continuing to type your homework in *Lurch*, because in the next chapter we'll find there is a great deal of value in being comfortable doing so.



[[⇒] [[and] [A] [B]] [C]]

Figure 4.4: Nested bubbles formed by placing the cursor in an inner bubble in expanded form

Quiz Yourself

- How do truth tables simplify the process of determining whether an argument is valid?
- Give a question that a one-line truth table is sufficient to answer.
- Give a question that a two-line truth table is sufficient to answer, but a one-line truth table is insufficient to answer.

Practice Exercises

While many of the exercises in this chapter have answers in the back, those answers do not include a justification by partial or full truth table. Be sure to include both an answer and a justification in your work.

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. $A \Rightarrow A$
2. $\neg B \wedge B$
3. $C \Rightarrow \neg C$
4. $\neg D \vee D$
5. $(A \Leftrightarrow B) \Leftrightarrow \neg(A \Leftrightarrow \neg B)$
6. $(A \wedge B) \vee (B \wedge A)$
7. $(A \Rightarrow B) \vee (B \Rightarrow A)$
8. $\neg(A \Rightarrow (B \Rightarrow A))$
9. $(A \wedge B) \Rightarrow (B \vee A)$
10. $A \Leftrightarrow (A \Rightarrow (B \wedge \neg B))$
11. $\neg(A \vee B) \Leftrightarrow (\neg A \wedge \neg B)$
12. $\neg(A \wedge B) \Leftrightarrow A$
13. $\big((A \wedge B) \wedge \neg(A \wedge B)\big) \wedge C$
14. $A \Rightarrow (B \vee C)$
15. $((A \wedge B) \wedge C) \Rightarrow B$
16. $(A \wedge \neg A) \Rightarrow (B \vee C)$
17. $\neg\big((C \vee A) \vee B\big)$
18. $(B \wedge D) \Leftrightarrow (A \Leftrightarrow (A \vee C))$

⋆ **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, $A \vee A$
3. $A \Rightarrow A$, $A \Leftrightarrow A$
4. $A \vee \neg B$, $A \Rightarrow B$
5. $A \wedge \neg A$, $\neg B \Leftrightarrow B$
6. $\neg(A \wedge B)$, $\neg A \vee \neg B$

7. $\neg(A \Rightarrow B)$, $\neg A \Rightarrow \neg B$
8. $(A \Rightarrow B)$, $(\neg B \Rightarrow \neg A)$
9. $((A \vee B) \vee C)$, $(A \vee (B \vee C))$
10. $((A \vee B) \wedge C)$, $(A \vee (B \wedge C))$

⋆ **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. $A \Rightarrow A$, $\neg A \Rightarrow \neg A$, $A \wedge A$, $A \vee A$
2. $A \wedge B$, $C \Rightarrow \neg B$, C
3. $A \vee B$, $A \Rightarrow C$, $B \Rightarrow C$
4. $A \Rightarrow B$, $B \Rightarrow C$, A, $\neg C$
5. $B \wedge (C \vee A)$, $A \Rightarrow B$, $\neg(B \vee C)$
6. $A \vee B$, $B \vee C$, $C \Rightarrow \neg A$
7. $A \Leftrightarrow (B \vee C)$, $C \Rightarrow \neg A$, $A \Rightarrow \neg B$
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. $A \Rightarrow A$, $\therefore A$
2. $A \vee \big(A \Rightarrow (A \Leftrightarrow A)\big)$, $\therefore$ A
3. $A \Rightarrow (A \wedge \neg A)$, $\therefore \neg A$
4. $A \Leftrightarrow \neg(B \Leftrightarrow A)$, $\therefore A$
5. $A \vee (B \Rightarrow A)$, $\therefore \neg A \Rightarrow \neg B$
6. $A \Rightarrow B$, B, $\therefore A$
7. $A \vee B$, $B \vee C$, $\neg A$, $\therefore B \wedge C$
8. $A \vee B$, $B \vee C$, $\neg B$, $\therefore A \wedge C$
9. $(B \wedge A) \Rightarrow C$, $(C \wedge A) \Rightarrow B$, $\therefore (C \wedge B) \Rightarrow A$
10. $A \Leftrightarrow B$, $B \Leftrightarrow C$, $\therefore A \Leftrightarrow C$

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

1. Suppose that $\mathcal{A}$ and $\mathcal{B}$ are logically equivalent. What can you say about $\mathcal{A} \Leftrightarrow \mathcal{B}$?
2. Suppose that $(\mathcal{A} \wedge \mathcal{B}) \Rightarrow \mathcal{C}$ is contingent. What can you say about the argument "$\mathcal{A}$, $\mathcal{B}$, $\therefore\mathcal{C}$"?
3. Suppose that $\{\mathcal{A}, \mathcal{B}, \mathcal{C}\}$ is inconsistent. What can you say about $(\mathcal{A} \wedge \mathcal{B} \wedge \mathcal{C})$?
4. Suppose that $\mathcal{A}$ is a contradiction. What can you say about the argument "$\mathcal{A}$, $\mathcal{B}$, $\therefore\mathcal{C}$"?
5. Suppose that $\mathcal{C}$ is a tautology. What can you say about the argument "$\mathcal{A}$, $\mathcal{B}$, $\therefore\mathcal{C}$"?
6. Suppose that $\mathcal{A}$ and $\mathcal{B}$ are logically equivalent. What can you say about $(\mathcal{A} \vee \mathcal{B})$?
7. Suppose that $\mathcal{A}$ and $\mathcal{B}$ are *not* logically equivalent. What can you say about $(\mathcal{A} \vee \mathcal{B})$?

Part F We could leave the biconditional ($\Leftrightarrow$) out of the language. If we did that, we could still write '$A \Leftrightarrow B$' so as to make sentences easier to read, but that would be shorthand for $(A \Rightarrow B) \wedge (B \Rightarrow A)$. The resulting language would be formally equivalent to SL, since $A \Leftrightarrow B$ and $(A \Rightarrow B) \wedge (B \Rightarrow A)$ 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 ($\Rightarrow$).

⋆ 1. $A \vee B$
⋆ 2. $A \wedge B$
⋆ 3. $A \Leftrightarrow B$

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. $A \wedge B$
5. $A \Rightarrow B$
6. $A \Leftrightarrow B$

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

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A}\|\mathcal{B}$
T	T	F
T	F	T
F	T	T
F	F	T

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 only the Sheffer stroke, write sentences that are equivalent to each of the following.

8. $\neg A$
9. $(A \wedge B)$
10. $(A \vee B)$
11. $(A \Rightarrow B)$
12. $(A \Leftrightarrow B)$

Chapter 5

Proofs in SL

Consider two arguments in SL:

Argument A

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

Argument B

$P \Rightarrow Q$
P
$\therefore$ Q

Clearly, these are valid arguments. You can confirm that they are valid by constructing four-line truth tables. Argument A makes use of an inference form that is always valid: Given a disjunction and the negation of one of the disjuncts, the other disjunct follows as a valid consequence. This rule is called *disjunctive syllogism.*

Argument B makes use of a different valid form: Given a conditional and its antecedent, the consequent follows as a valid consequence. This is called *modus ponens.*

The method of truth tables is useful for showing the validity of these arguments, but it does not clearly show *why* they are 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 provided you made no mistakes in constructing the table, then you would know that the argument was valid. Yet you would not be able to say anything further about *why* that particular argument was a valid argument form.

The aim of a *proof system* is to show that particular arguments are valid in a way that allows us to understand the reasoning involved in the argument. A proof system begins with basic argument forms, like disjunctive syllogism and modus ponens. These forms can then be combined to make more complicated arguments, like this one:

(1) $\neg L \Rightarrow (J \vee L)$
(2) $\neg L$
$\therefore$ J

By modus ponens, (1) and (2) entail $J \vee L$. This is an *intermediate conclusion.* It follows logically from the premises, but it is not the conclusion we want. Now $J \vee L$ and (2) entail J, by disjunctive syllogism. We do not need to call the above example a new argument form. The proof of the argument shows that it is really just a combination of rules we have already introduced.

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

This chapter covers a proof system for SL.

5.1 Basic rules for SL

In designing a proof system, we could just start with disjunctive syllogism and modus ponens. Whenever we discovered a valid argument which could not be proven with rules we already had, we could introduce it as a new rule. Proceeding in this way, we would have an unsystematic grab bag of rules. We might accidently add some strange rules, and we might end up with more rules than we need.

Instead, we will develop what is called a NATURAL DEDUCTION system. In a natural deduction system, there will be two rules for each logical operator: an INTRODUCTION rule that allows us to prove a sentence that has that operator as the main connective and an ELIMINATION rule that allows us to prove something from a sentence that has that operator as the main connective.

In addition to the rules for each logical operator, we will also have a reiteration rule. If you already have shown something in the course of a proof, the reiteration rule allows you to repeat it on a new line. For instance:

1. $\mathcal{A}$
2. $\mathcal{A}$ R 1.

When we add a line to a proof, we write the rule that justifies that line. We also write the numbers of the lines to which the rule was applied. A use of the reiteration rule is justified by one line, the line that you are reiterating. So the 'R 1' on line 2 of the proof means that line 2 is justified by the reiteration rule (R) applied to line 1.

Obviously, the reiteration rule will not allow us to show anything *new*. For that, we will need more rules. The remainder of this section will give introduction and elimination rules for all of the logical connectives. This will give us a complete proof system for SL.

All of the rules introduced in this chapter are summarized starting on p. 202.

Conjunction

Think for a moment: What would you need to show in order to prove $E \wedge F$?

Of course, you could show $E \wedge F$ by proving E and separately proving F. This holds even if the two conjuncts are not atomic sentences. If you can prove $((A \vee J) \Rightarrow V)$ and $((V \Rightarrow L) \Leftrightarrow (F \vee N))$, then you have effectively proven

$$((A \vee J) \Rightarrow V) \wedge ((V \Rightarrow L) \Leftrightarrow (F \vee N)).$$

So this will be our conjunction introduction rule, which we abbreviate $\wedge$I:

m. $\mathcal{A}$

n. $\mathcal{B}$

$\mathcal{A} \wedge \mathcal{B}$ $\wedge$I m., n.

A line of proof must be justified by some rule, and here we have '$\wedge$I *m,n*.' This means: Conjunction introduction applied to line m and line n. These are variables, not real line numbers; m is some line and n is some other line. In an actual proof, the lines are numbered $1., 2., 3., \ldots$ and rules must be applied to specific line numbers. When we define the rule, however, we use variables to underscore the point that the rule may be applied to any two lines that are already in the proof. If you have K on line 8 and L on line 15, you can prove $K \wedge L$ at some later point in the proof with the justification '$\wedge$I 8, 15.'

Now, consider the elimination rule for conjunction. What are you entitled to conclude from a sentence like $E \wedge F$? Surely, you are entitled to conclude E; if $E \wedge F$ were true, then E would be true. Similarly, you are entitled to conclude F. This will be our conjunction elimination rule, which we abbreviate $\wedge$E:

$m.$ $\quad \mathcal{A} \wedge \mathcal{B}$

$\quad \mathcal{A}$ $\qquad$ $\wedge$E m.

$\quad \mathcal{B}$ $\qquad$ $\wedge$E m.

When you have a conjunction on some line of a proof, you can use $\wedge$E to derive either of the conjuncts. The $\wedge$E rule requires only one sentence, so we write one line number as the justification for applying it.

Even with just these two rules, we can provide some proofs. Consider this argument.

$$\begin{array}{rl} & ((A \vee B) \Rightarrow (C \vee D)) \wedge ((E \vee F) \Rightarrow (G \vee H)) \\ \therefore & ((E \vee F) \Rightarrow (G \vee H)) \wedge ((A \vee B) \Rightarrow (C \vee D)) \end{array}$$

The main logical operator in both the premise and conclusion is conjunction. Since conjunction is symmetric, the argument is obviously valid. In order to provide a proof, we begin by writing down the premise, so the beginning of the proof looks like this:

1. $\quad ((A \vee B) \Rightarrow (C \vee D)) \wedge ((E \vee F) \Rightarrow (G \vee H))$

From the premise, we can get each of the conjuncts by $\wedge$E. The proof now looks like this:

1. $\quad ((A \vee B) \Rightarrow (C \vee D)) \wedge ((E \vee F) \Rightarrow (G \vee H))$
2. $\quad (A \vee B) \Rightarrow (C \vee D)$ $\qquad$ $\wedge$E 1.
3. $\quad (E \vee F) \Rightarrow (G \vee H)$ $\qquad$ $\wedge$E 1.

We can tell that the first line is a premise because it has no reason to support it. It would also be acceptable to write 'given' as its reason, as many high-school geometry texts do.

The rule $\wedge$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 $\wedge$I rule to lines 3 and 2, we arrive at the desired conclusion. The finished proof looks like this:

1. $\quad ((A \vee B) \Rightarrow (C \vee D)) \wedge ((E \vee F) \Rightarrow (G \vee H))$
2. $\quad (A \vee B) \Rightarrow (C \vee D)$ $\qquad$ $\wedge$E 1.
3. $\quad (E \vee F) \Rightarrow (G \vee H)$ $\qquad$ $\wedge$E 1.
4. $\quad ((E \vee F) \Rightarrow (G \vee H)) \wedge ((A \vee B) \Rightarrow (C \vee D))$ $\qquad$ $\wedge$I 3., 2.

This proof is trivial, but it shows how we can use rules of proof together to demonstrate the validity of an argument form. Also, 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.

Quiz Yourself

- What are the English names of the rules for which we use the symbols $\wedge$I and $\wedge$E?
- The premise in any use of the $\wedge$E rule must take what form?
- Given two distinct premises, how many different conclusions are possible with one use of the $\wedge$I rule?

Disjunction

If M were true, then $M \vee N$ would also be true. So the disjunction introduction rule ($\vee$I) allows us to derive a disjunction if we have one of the two disjuncts:

m.	$\mathcal{A}$	
	$\mathcal{A} \vee \mathcal{B}$	$\vee$I m.
	$\mathcal{B} \vee \mathcal{A}$	$\vee$I m.

Notice that $\mathcal{B}$ can be *any* sentence whatsoever. So the following is a legitimate proof:

1.	M	
2.	$M \vee (((A \Leftrightarrow B) \Rightarrow (C \wedge D)) \Leftrightarrow (E \wedge F))$	$\vee$I 1.

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 $\mathcal{A}$ is true, then $\mathcal{A} \vee \mathcal{B}$ is true regardless of what $\mathcal{B}$ is. So the conclusion could not be false if the premise were true; the argument is valid.

Now consider the disjunction elimination rule. What can you conclude from $M \vee N$? You cannot conclude M. It might be M's truth that makes $M \vee N$ true, as in the example above, but it might not. From $M \vee N$ alone, you cannot conclude anything about either M or N specifically. If you also knew that N was false, however, then you would be able to conclude M.

This is just disjunctive syllogism, and it will be the disjunction elimination rule ($\vee$E).

m.	$\mathcal{A} \vee \mathcal{B}$		m.	$\mathcal{A} \vee \mathcal{B}$	
n.	$\neg\mathcal{B}$		n.	$\neg\mathcal{A}$	
	$\mathcal{A}$	$\vee$E m., n.		$\mathcal{B}$	$\vee$E m., n.

Conditional

Consider this argument:

$$\begin{array}{rl} & R \vee F \\ \therefore & \neg R \Rightarrow F \end{array}$$

The argument is certainly a valid one. 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, like this:

$$\begin{array}{ll} 1. & R \vee F \end{array}$$

If we had $\neg R$ as a further premise, we could derive F by the $\vee$E rule. 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 indent deeper to indicate that we are no longer in the main proof. Then we write an assumption for the subproof. This can be anything we want. Here, it will be helpful to assume $\neg R$. Our proof now looks like this:

$$\begin{array}{lll} 1. & R \vee F & \\ 2. & & \neg R \end{array}$$

It is important to notice that we are not claiming to have proven $\neg R$. We do not need to write in any justification for the assumption line of a subproof, just as we did not for our original premise. You can think of the subproof as posing the question: What could we show *if* $\neg R$ were true? For one thing, we can derive F. So we do:

$$\begin{array}{llll} 1. & R \vee F & & \\ 2. & & \neg R & \\ 3. & & F & \vee\text{E } 1., 2. \end{array}$$

This has shown that *if* we had $\neg R$ as a premise, *then* we could prove F. In effect, we have proven $\neg R \Rightarrow F$.

Notice that, because we are reasoning from the assumption R, we indent the line containing F. We *cannot* conclude F with no indentation, as if we knew F in the main proof, because F follows only from the assumption of R. Once you have started a subproof, you must stay in it until you use a rule that allows you to close the subproof.

The conditional introduction rule ($\Rightarrow$I) is just such a rule; it will allow us to close the subproof (thus unindenting) and derive $\neg R \Rightarrow F$ in the main proof. Our final proof therefore looks like this:

$$\begin{array}{llll} 1. & R \vee F & & \\ 2. & & \neg R & \\ 3. & & F & \vee\text{E } 1., 2. \\ 4. & \neg R \Rightarrow F & & \Rightarrow\text{I } 2., 3. \end{array}$$

Notice that the justification for applying the $\Rightarrow$I rule is the entire subproof, cited by its starting and ending lines. Usually the subproof will contain more than just two lines, but this is a small first example.

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:

1.	$\mathcal{A}$	
2.	$\quad\mathcal{B}$	
3.	$\quad\mathcal{B}$	R 2.

It may seem as if this is a proof that you can derive any conclusion $\mathcal{B}$ from any premise $\mathcal{A}$. In order to complete a proof, you must *close* all of the subproofs, that is, draw from them a conclusion that follows outside the subproof. Once you close a subproof, you cannot refer back to individual lines inside it. Therefore we cannot close the subproof and then use the R rule on line 4 to derive $\mathcal{B}$ in the main proof.

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 besides the original premises of the argument.

Of course, it is legitimate to do this:

1.	$\mathcal{A}$	
2.	$\quad\mathcal{B}$	
3.	$\quad\mathcal{B}$	R 2.
4.	$\mathcal{B} \Rightarrow \mathcal{B}$	$\Rightarrow$I 2., 3.

This should not seem so strange, though. Since $\mathcal{B} \Rightarrow \mathcal{B}$ is a tautology, no particular premises should be required to validly derive it. (Indeed, as we will see, a tautology follows from any premises.)

Put in a general form, the $\Rightarrow$I rule looks like this:

$m.$	$\quad\mathcal{A}$	want $\mathcal{B}$
$n.$	$\quad\mathcal{B}$	
	$\mathcal{A} \Rightarrow \mathcal{B}$	$\Rightarrow$I $m.$, $n.$

When you introduce a subproof, it is helpful to write what you want to derive in the right-hand column. 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; this is just 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 an arbitrary, wacky assumption would just waste lines of the proof. In order to derive a conditional by the $\Rightarrow$I, for instance, you must assume the antecedent of the conditional in a subproof.

The $\Rightarrow$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.

Now consider the conditional elimination rule. Nothing follows from $M \Rightarrow N$ alone, but if we have both $M \Rightarrow N$ and M, then we can conclude N. This rule, modus ponens, will be the conditional elimination rule ($\Rightarrow$E).

$m.$	$\mathcal{A} \Rightarrow \mathcal{B}$	
$n.$	$\mathcal{A}$	
	$\mathcal{B}$	$\Rightarrow$E $m.$, $n.$

Now that we have rules for the conditional, consider this argument:

$$\begin{array}{rl} & P \Rightarrow Q \\ & Q \Rightarrow R \\ \therefore & P \Rightarrow R \end{array}$$

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 $\Rightarrow$I rule. For that, we need a subproof— so we write in the antecedent of the conditional as assumption of a subproof:

1.	$P \Rightarrow Q$	
2.	$Q \Rightarrow R$	
3.		P

We made P available by assuming it in a subproof, allowing us to use $\Rightarrow$E on the first premise. (Earlier lines in the main proof are usable in the subproof, but not the other way around.) This gives us Q, which allows us to use $\Rightarrow$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 $\Rightarrow$I rule and finish the proof.

1.	$P \Rightarrow Q$		
2.	$Q \Rightarrow R$		
3.		P	want R
4.		Q	$\Rightarrow$E 1., 3.
5.		R	$\Rightarrow$E 2., 4.
6.	$P \Rightarrow R$		$\Rightarrow$I 3., 5.

Quiz Yourself

- To conclude the statement $R \vee S$ using $\vee$I, what premises might we use?
- When beginning a subproof, are you permitted to assume any SL sentence at all?
- Is it necessary to end a subproof before you end the proof that contains it?

Biconditional

The rules for the biconditional will be like double-barreled versions of the rules for the conditional.

In order to derive $W \Leftrightarrow X$, for instance, you must know that X implies W *and* that W implies X. Thus the biconditional introduction rule ($\Leftrightarrow$I) requires two conditionals to be established.

$m.$	$\mathcal{A} \Rightarrow \mathcal{B}$	
$n.$	$\mathcal{B} \Rightarrow \mathcal{A}$	
	$\mathcal{A} \Leftrightarrow \mathcal{B}$	$\Leftrightarrow$I $m.$, $n.$

The biconditional elimination rule ($\Leftrightarrow$E) lets you do a bit more than the conditional rule. 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:

$m.$	$\mathcal{A} \Leftrightarrow \mathcal{B}$		$m.$	$\mathcal{A} \Leftrightarrow \mathcal{B}$	
$n.$	$\mathcal{A}$		$n.$	$\mathcal{B}$	
	$\mathcal{B}$	$\Leftrightarrow$E $m.$, $n.$		$\mathcal{A}$	$\Leftrightarrow$E $m.$, $n.$

Negation

Here is a simple mathematical argument in English:

Assume there is some largest whole number. Call it A.
Then $A + 1$ is also a whole number.
Obviously, $A + 1 > A$.
So there is a whole number greater than A.
This is impossible, since A is assumed to be the largest whole number.
$\therefore$ The assumption was wrong: There is no largest whole number.

This argument form is traditionally called a *reductio*. Its full Latin name is *reductio ad absurdum*, which means 'reduction to absurdity.' In a reductio, we assume something for the sake of argument— for example, that there is a largest whole number. Then we show that the assumption leads to two contradictory sentences— for example, that A is the largest whole number and that it is not. In this way, we show that the original assumption must have been false, because its consequences make no sense— they are 'absurd.'

The basic rules for negation enable arguments like this. 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 ($\neg$I) rule:

$m.$	$\mathcal{A}$	for reductio
$n.$	$\mathcal{B}$	
$p.$	$\neg\mathcal{B}$	
	$\neg\mathcal{A}$	$\neg$I $m.$, $n.$, $p.$

For the rule to apply, the two lines marked n and p must be an explicit contradiction: some sentence and its negation. We write 'for reductio' as a note to ourselves, a reminder of why we started the subproof. It is not formally part of the proof, and you can leave it out if you find it distracting.

To see how the rule works, suppose we want to prove the law of non-contradiction: $\neg(G \wedge \neg G)$. We can prove this without any premises by immediately starting a subproof. We want to apply $\neg$I to the subproof, so we assume $G \wedge \neg G$. We then get an explicit contradiction by $\wedge$E. The proof looks like this:

1.	$G \wedge \neg G$	for reductio
2.	G	$\wedge$E 1.
3.	$\neg G$	$\wedge$E 1.
4.	$\neg(G \wedge \neg G)$	$\neg$I 1., 2., 3.

The $\neg$E rule will work in much the same way. If we assume $\neg\mathcal{A}$ and show that it leads to a contradiction, we have effectively proven $\mathcal{A}$. So the rule looks like this:

$m.$	$\neg\mathcal{A}$	for reductio
$n.$	$\mathcal{B}$	
$p.$	$\neg\mathcal{B}$	
	$\mathcal{A}$	$\neg$E $m.$, $n.$, $p.$

Quiz Yourself

- Why were the negation rules called *reductio ad absurdum?*
- Which of the two negation rules can draw the one-sentence-letter conclusion P?
- In that same situation, what assumption would be made to start the subproof?

5.2 Derived rules

The rules of the natural deduction system are meant to be systematic. There is an introduction and an elimination rule for each logical operator, but why these basic rules rather than some others? Many natural deduction systems have a disjunction elimination rule that works like this:

$m.$	$\mathcal{A} \vee \mathcal{B}$	
$n.$	$\mathcal{A} \Rightarrow \mathcal{C}$	
$p.$	$\mathcal{B} \Rightarrow \mathcal{C}$	
	$\mathcal{C}$	$\vee *$ $m.$, $n.$, $p.$

It might seem as if there will be some proofs that we cannot do with our proof system, because we do not have this rule. Yet this is not the case. If you can do a proof with this rule, you can do a proof with the basic rules of the natural deduction system. Consider this proof:

1.	$\mathcal{A} \vee \mathcal{B}$			
2.	$\mathcal{A} \Rightarrow \mathcal{C}$			
3.	$\mathcal{B} \Rightarrow \mathcal{C}$			want $\mathcal{C}$
4.		$\neg\mathcal{C}$		for reductio
5.			$\mathcal{A}$	for reductio
6.			$\mathcal{C}$	$\Rightarrow$E 2., 5.
7.			$\neg\mathcal{C}$	R 4.
8.		$\neg\mathcal{A}$		$\neg$I 5., 6., 7.
9.			$\mathcal{B}$	for reductio
10.			$\mathcal{C}$	$\Rightarrow$E 3., 9.
11.			$\neg\mathcal{C}$	R 4.
12.		$\neg\mathcal{B}$		$\neg$I 9., 10., 11.
13.		$\mathcal{B}$		$\vee$E 1., 8.
14.	$\mathcal{C}$			$\neg$E 4., 13., 12.

$\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ are 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. It provides a pattern that can prove anything that the $\vee*$ rule can prove, using only the basic rules of SL. This means that $\vee*$ is not really necessary. Adding it to the list of basic rules would not allow us to derive anything that we could not derive without it.

Nevertheless, the $\vee*$ rule would be convenient. It would allow us to do in one line what requires eleven lines and several nested subproofs with the basic rules. So we will add $\vee*$ to the proof system as a derived rule.

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. Anytime you use the $\vee*$ rule, you could always take ten extra lines and prove the same thing without it.

For the sake of convenience, we will add several other derived rules. One is *modus tollens* (MT).

m.	$\mathcal{A} \Rightarrow \mathcal{B}$	
n.	$\neg\mathcal{B}$	
	$\neg\mathcal{A}$	MT m., n.

We leave the proof of this rule as an exercise. Note that if we had already proven the MT rule, then the proof of the $\vee*$ rule could have been done in only five lines.

We also add hypothetical syllogism (HS) as a derived rule. We have already given a proof of it on p. 57.

m.	$\mathcal{A} \Rightarrow \mathcal{B}$	
n.	$\mathcal{B} \Rightarrow \mathcal{C}$	
	$\mathcal{A} \Rightarrow \mathcal{C}$	HS m., n.

5.3 Derived rules of logical equivalence

There are many derived rules that come in the form of logical equivalences. Here are a few that express the commutativity of several logical connectives. We abbreviate these derived rules as 'Comm.' Each is a single sentence of SL that can be added to any proof, using Comm as the reason, and no premises.

$$\begin{array}{rl} (\mathcal{A} \wedge \mathcal{B}) \Leftrightarrow (\mathcal{B} \wedge \mathcal{A}) & \\ (\mathcal{A} \vee \mathcal{B}) \Leftrightarrow (\mathcal{B} \vee \mathcal{A}) & \\ (\mathcal{A} \Leftrightarrow \mathcal{B}) \Leftrightarrow (\mathcal{B} \Leftrightarrow \mathcal{A}) & \quad \text{Comm} \end{array}$$

Another derived rule of logical equivalence is double negation (DN). This is the rule:

$$\neg\neg\mathcal{A} \Leftrightarrow \mathcal{A} \quad \text{DN}$$

Two more such rules are called De Morgan's Laws, named for the 19th-century British logician August De Morgan. (Although De Morgan did discover these laws, he was not the first to do so.) The rules capture useful relations between negation, conjunction, and disjunction. Here are the rules, which we abbreviate DeM:

$$\begin{array}{rl} \neg(\mathcal{A} \vee \mathcal{B}) \Leftrightarrow (\neg\mathcal{A} \wedge \neg\mathcal{B}) & \\ \neg(\mathcal{A} \wedge \mathcal{B}) \Leftrightarrow (\neg\mathcal{A} \vee \neg\mathcal{B}) & \quad \text{DeM} \end{array}$$

Because $\mathcal{A} \Rightarrow \mathcal{B}$ is a *material conditional*, it is equivalent to $\neg\mathcal{A} \vee \mathcal{B}$. A further derived rule captures this equivalence. We abbreviate the rule MC, for 'material conditional.' It takes two forms:

$$\begin{array}{rl} (\mathcal{A} \Rightarrow \mathcal{B}) \Leftrightarrow (\neg\mathcal{A} \vee \mathcal{B}) & \\ (\mathcal{A} \vee \mathcal{B}) \Leftrightarrow (\neg\mathcal{A} \Rightarrow \mathcal{B}) & \quad \text{MC} \end{array}$$

A final rule captures the relation between conditionals and biconditionals. We will call this rule biconditional exchange and abbreviate it ⇔ex.

$$((\mathcal{A} \Rightarrow \mathcal{B}) \wedge (\mathcal{B} \Rightarrow \mathcal{A})) \Leftrightarrow (\mathcal{A} \Leftrightarrow \mathcal{B}) \quad \Leftrightarrow\text{ex}$$

Quiz Yourself

- What is the difference between a basic rule and a derived rule?
- You almost never see two adjacent ¬ symbols in a wff of SL. Why not?

Practice Exercises

Part A Provide a justification (rule and line numbers) for each line of proof that requires one. Although it is not necessary to give reasons for proof premises, I have marked them 'given' here because in this exercise, *all* reasons were left blank, and thus you need some other way to distinguish the premises from the rest of the proof.

1.	$W \Rightarrow \neg B$	given
2.	$A \wedge W$	given
3.	$B \vee (J \wedge K)$	given
4.	W	
5.	$\neg B$	
6.	$J \wedge K$	
7.	K	

1.	$L \Leftrightarrow \neg O$	given
2.	$L \vee \neg O$	given
3.	$\quad \neg L$	
4.	$\quad \neg O$	
5.	$\quad L$	
6.	$\quad \neg L$	
7.	L	

1.	$F \Rightarrow (G \wedge H)$	given
2.	$\quad F$	
3.	$\quad G \wedge H$	
4.	$\quad G$	
5.	$F \Rightarrow G$	

1.	$Z \Rightarrow (C \wedge \neg N)$	given
2.	$\neg Z \Rightarrow (N \wedge \neg C)$	given
3.	$\quad \neg(N \vee C)$	
4.	$\quad \neg N \wedge \neg C$	
5.	$\quad\quad Z$	
6.	$\quad\quad C \wedge \neg N$	
7.	$\quad\quad C$	
8.	$\quad\quad \neg C$	
9.	$\quad \neg Z$	
10.	$\quad N \wedge \neg C$	
11.	$\quad N$	
12.	$\quad \neg N$	
13.	$N \vee C$	

Part B Type the four proofs from Part A into *Lurch*, as a way to learn how to format proofs correctly. Here are some requirements and tips.

1. Be sure to use numbered lists (the toolbar button next to indent and unindent) as an automatic way to number the lines in the proof.
2. In the preferences window, be sure the choice for 'Interpret tabs as indentation changes' is unchecked, so that the tab key does not indent line numbers and create sublists, but allows you to indent the sentences themselves instead. Then be sure to indent each line properly.
3. Also use tabs to line up the column of reasons to the right of each line's logical expression.
4. Each line's sentence should be marked as meaningful, as you learned to do in Section 3.5. Ensure that in the bubble tag, *Lurch* classifies the expression as the type you expected. (If any are marked 'string,' that means that *Lurch* can't understand it, and is only seeing a meaningless string of symbols, so you should find and fix your typographical error before proceeding.)

In the next chapter we'll see how to get *Lurch* to check proofs that you type in this way, and then I'll assign you to do proofs on your own.

Chapter 6

Proofs in *Lurch*

6.1 Having *Lurch* check your work

In order for *Lurch* to read a proof and give you feedback about whether it's constructed correctly, you must tell *Lurch* the structure of each line in the proof, in three ways.

1. You must put the statement of the proof line inside a red bubble, which tells *Lurch* that it is a Meaningful Expression (ME). You already know how to do this, and practiced it in the exercises from the previous chapter.
2. You must put the reason of the proof line inside a blue Reason bubble, which tells *Lurch* what reason you're using to support the corresponding statement. I'll show you below how to do this.
3. You must put any premise numbers cited in the proof line inside blue Premise bubbles, which tell *Lurch* what premises you're citing in support of the line. I'll show you below how to do this as well.

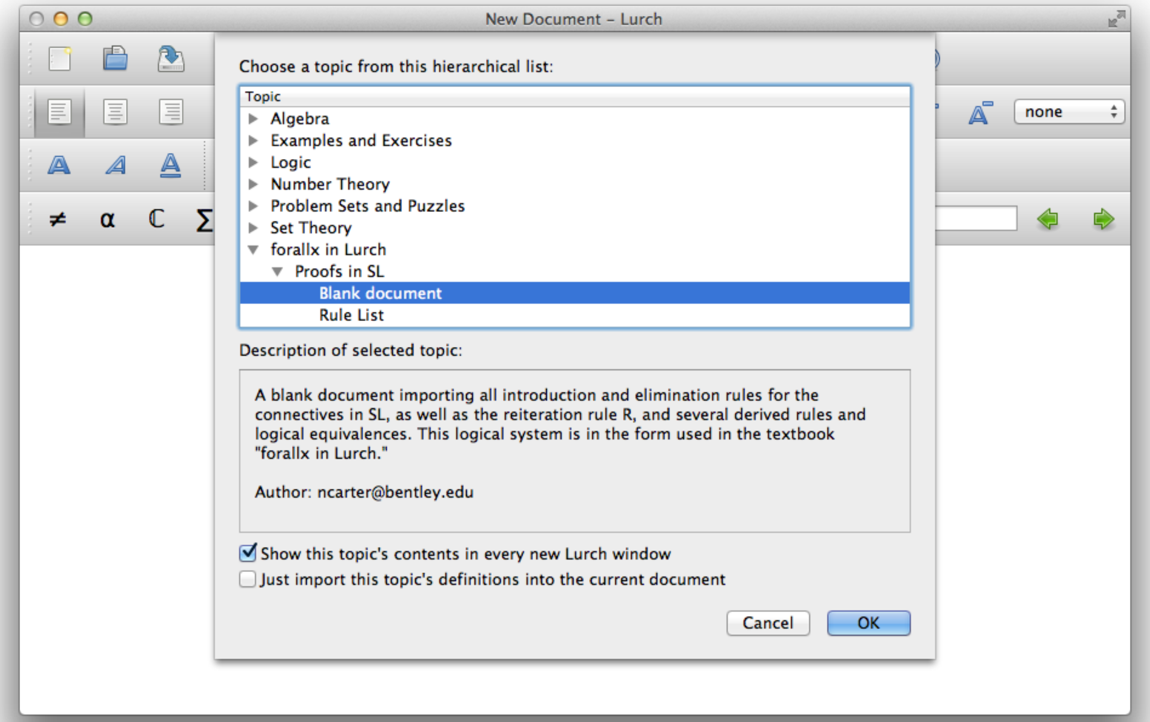
I suggest you follow along in *Lurch* as you read this section. I introduce a very simple document below, and show you how to add Reason and Premise bubbles to it. Do so in *Lurch* on your computer, so that you're confident that you know how to use these skills in this chapter's practice problems.

Choosing the correct topic

Before beginning, you must tell *Lurch* which mathematical topic you'll be working in, so that it knows what kinds of rules you'll be using. Later in this text, you will have learned far more rules than you have now, and *Lurch* knows them all. So we must tell *Lurch* to import just the rules we know so far, to keep us from encountering content we haven't yet learned. (There are also branches of mathematics very different from logic, and we don't need *Lurch* to import their rules right now either.)

After opening *Lurch*, go to the File menu, and click "Choose topic..." From the hierarchical list of topics that appears, choose the category named after this textbook, and the subcategory "Proofs in SL," and then choose "Blank document," as illustrated in Figure 6.1. Just as shown in that figure, ensure that only the first of the two checkboxes at the bottom of the window is checked, then click OK.

The result should be a blank document, but it's a blank document that knows all the rules of SL you learned in the previous chapter, and will let you use them.



Choose a topic from this hierarchical list:

- Algebra
- Examples and Exercises
- Logic
- Number Theory
- Problem Sets and Puzzles
- Set Theory
- forallx in Lurch
  - Proofs in SL
    - Blank document
    - Rule List

Description of selected topic:

A blank document importing all introduction and elimination rules for the connectives in SL, as well as the reiteration rule R, and several derived rules and logical equivalences. This logical system is in the form used in the textbook "forallx in Lurch."

Author: ncarter@bentley.edu

[✓] Show this topic's contents in every new Lurch window
[ ] Just import this topic's definitions into the current document

Figure 6.1: Choosing the correct topic for beginning proofs in SL in *Lurch*

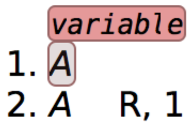


Figure 6.2: The short proof from page 52, typed into *Lurch*

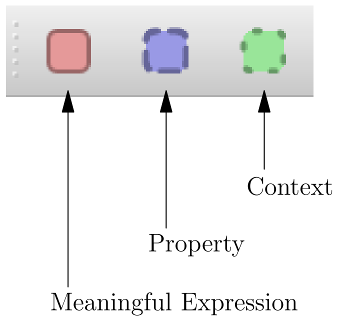


Figure 6.3: The three buttons on the *Lurch* toolbar for inserting bubbles into a document, and the purposes of each. The Meaning menu contains these same three actions, with corresponding keyboard shortcuts, for users who prefer that interface.

Marking statements

I begin with a very small example proof, which appeared on page 52 of this text. I have typed the proof into *Lurch*, using a numbered list to provide line numbers, and the result is shown in Figure 6.2. Each statement is surrounded in a red ME bubble, and you can see the first bubble in that figure, because the cursor was inside the bubble at the time the image was captured. The tag above the bubble says "variable" because a single propositional letter is sometimes called a propositional variable.

Type now this same two-line proof into the blank document you just opened in *Lurch*, and surround each statement in a separate ME bubble. Once you have done so, you have accomplished the first item on the list of three from page 63. Once we do the other two, *Lurch* will see the full structure of the proof, and will be able to tell us whether it is valid.

Reason bubbles

The next step is to mark all the reasons in the proof. In this proof, there is only one, the R on the second line. Select the R so that we can wrap it in a Reason bubble. Reason bubbles are one of several kinds of Property bubbles, which attach information to a nearby ME. We'd like to attach the reason R to the statement in line 2, so we will make the R a Reason Property of that statement.

The three toolbar buttons for creating bubbles in *Lurch* are shown in Figure 6.3; so far you've only used the leftmost one, for marking expressions as meaningful. Now, with the R selected, click the blue button, for making it a Property. The result should look like Figure 6.4, with your cursor in the new bubble.

The tag on the blue bubble indicates two things. First, its arrow indicates that it modifies the ME to its left, the A in line 2. You can change that arrow to modify more MEs to the left, or to modify one or more MEs to the right, by clicking the arrow itself with your left or right mouse button. Feel free to experiment with doing so, but we do not need to change the arrow for this proof; we want the R to modify the statement A

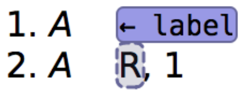


Figure 6.4: The same proof as in Figure 6.2, but now with the R marked as a property. This is not yet the correct way to mark R as a reason for line 2, but it is the first step in the process.

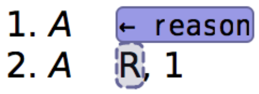


Figure 6.5: The same proof as in Figure 6.4, but now with the R correctly marked as a Reason.

on line 2. Second, the bubble tag says "label" because it means that R is acting as a Label on the statement A. This is not what we want; we want the R to act as the *Reason* for the statement A. Property bubbles default to being Labels, but we have no need for Labels yet in our work, so we need to change this.

To do so, click the word "label" in the bubble tag, and from the list of choices that appear, choose Reason instead. Your blue bubble should then look like the one in Figure 6.5. Now step 2 in the list on page 63 is complete.

Premise bubbles

The final element of the proof is that on line 2 we have not only cited R as a reason, but have cited line 1 as the premise required by the R rule. So we must tell *Lurch* that the number 1 is not just meaningless text, but its meaning is a Premise citation.

To do so, proceed just as you did for marking R as a reason. Highlight the number 1 that follows the R, click the blue Property button shown in Figure 6.3, and then click the bubble tag to change it from a Label to a Premise. The result should look like Figure 6.6.

Automatic validation

The cornerstone of *Lurch* is its ability to check your proofs. Now, this first proof we've typed into *Lurch* is a tiny example, and I think we're all pretty sure that it's correct! But once you start doing larger proofs, especially ones you've constructed from scratch, it's very valuable to have an automatic tutor looking over your shoulder, giving positive feedback for your correct steps and telling you to stop and fix something when you make a mistake. This prevents your practicing bad habits, and helps you feel more confident in developing good ones. So let's see how to get *Lurch* to check this simple proof.

Lurch uses colored traffic lights or thumbs up and down to give feedback on your work, as shown in Figure 6.7. Once we ask *Lurch* to check our work on this simple proof, we should expect a colored traffic light for

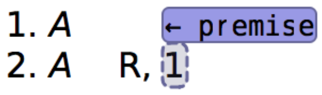


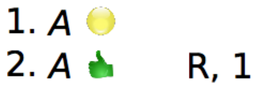
Figure 6.6: The same proof as in Figure 6.5, but now with the 1 correctly marked as a Premise.

🟢 👍 A green traffic light or a green thumbs-up indicates a correct step of work.

🔴 👎 A red traffic light or a red thumbs-down indicates an incorrect step of work.

🟡 A yellow traffic light indicates an undischarged assumption.

Figure 6.7: The colored traffic light icons *Lurch* uses to give feedback on each step of your work. For colorblind users or when printing your documents in black and white, there is an option in the *Lurch* preferences to use thumbs-up and thumbs-down icons instead. Throughout this text, I use the thumb icons, to support readability on black and white printers and devices.



1. A 🟡
2. A 👍 R, 1

Figure 6.8: The proof from Figure 6.6, now with validation enabled. See Section 6.1 for explanations of the yellow and green lights and thumbs.

each line of the proof, telling us we're correct.

To turn on validation of your work, click the toolbar button that looks just like the green traffic light in Figure 6.7. The result should look just like Figure 6.8. But why is one of the results yellow? Isn't our whole proof correct?

The green thumbs-up in Figure 6.8 (which may be a green light on your computer, depending on your settings) tells us that the one step of work that we actually justified with a reason and a premise is valid. The yellow light indicates that the first line has no reason.

A real yellow traffic light on the road doesn't mean that you must stop, but rather that you should slow down and consider *whether* you need to stop. A yellow traffic light in *Lurch* functions the same way. It does not mean that the first line in the proof is wrong, but it alerts us to the fact that we didn't justify it.

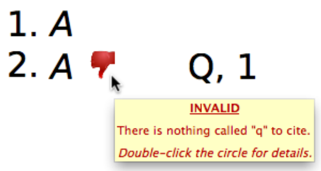
That may be okay. In this proof, for example, it *is* okay—the intent of this proof is to begin with the assumption A and use one tiny step of logic to conclude A again from it. We *want* the first line of the proof to be an unjustified assumption. But the yellow light just makes sure you've noticed that line 1 is unjustified, so you can check to be sure that's what you intended.

So our first, tiny *Lurch* proof checks out just as we had hoped! But what if we had done something wrong? Let's see how *Lurch* would have responded.

Try changing something in line 2 of the proof. For example, you might change the A to a B, or you might change the R to a J, or you might change the 1 to a 7. *Lurch*'s response is that all traffic lights disappear, which may not be what you expected.

Lurch removes all validation indicators whenever you change something in your document. To have it check your work again, click the green light on the toolbar again. If you would rather *Lurch* check your proofs continually as you work, you can click the lock button next to the green traffic light on the toolbar, to lock validation on. In large proofs, locking validation on can slow down the software, so use that feature judiciously.

Once you have made a change and re-enabled validation, you should see a red traffic light (or thumbs-down)


1. A
2. A Q, 1

INVALID
There is nothing called "q" to cite.
Double-click the circle for details.

Figure 6.9: The proof from Figure 6.8, now with a mistake introduced. The mouse is hovering over the red validation icon so that *Lurch* provides additional feedback about *why* the step was judged invalid.

on line 2. But just knowing that the step is wrong is not nearly as helpful as knowing why. Hover your mouse over the red traffic light or thumbs-down to see why *Lurch* judged it incorrect. You should get a message like the one in Figure 6.9, depending on which mistake you chose to introduce.

In my case, I chose to cite a rule that doesn't exist, so *Lurch* told me so. It says that I can double-click the circle (i.e., the traffic light icon—or a thumbs-down in this case) for more information. Doing so pops up a window full of information on how *Lurch* determined the step to be invalid, sometimes more information than you want! But it's good to know that information is there for those times when the small yellow box of hover text doesn't tell you enough. Also, that detailed feedback usually contains suggestions on how to fix the error.

Thus *Lurch* can grade your proofs as you do them, which is a great benefit! You do not need to wait several days after handing in an assignment (until your instructor has been able to grade it and another class meeting has come around) to find out if you are learning and using logic correctly. You can find out mere moments after writing a proof line whether it is correct, in the comfort of your own home, dormitory, office, or anywhere you have a computer.

Quiz Yourself

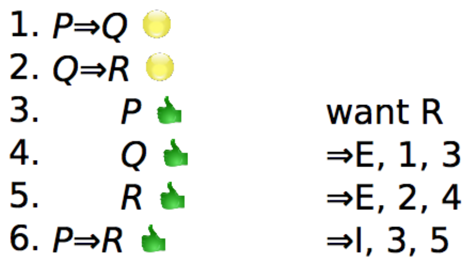
- What are the three pieces of information you need to give to *Lurch* to have it grade a step in a proof?
- What types of proof steps do not require premises or reasons attached to them?
- What does *Lurch* mean when it puts a yellow traffic light next to a wff in a proof?

Details about using *Lurch* to write proofs

The rule list. To find out which rules of logic *Lurch* knows about, and the names by which you should cite them, see the Meaning menu, which contains an item called "List all defined rules." It shows you a table of all the rules currently defined for your use, and the names by which you should cite them.

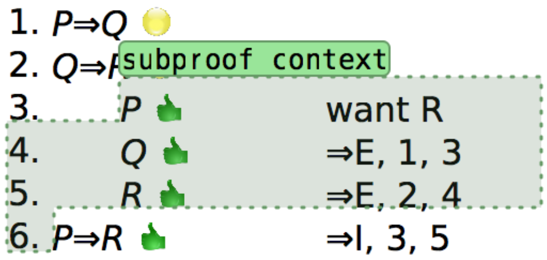
You should find them named exactly the same in *Lurch* as they are in this text. *Lurch* does not care about capitalization in rule names, so for example you could cite the R rule as r instead.

Subproofs. Consider the bubbled and validated proof in Figure 6.10. (It is one of the exercises from the previous chapter.) It has a subproof that begins on line 3 and extends to line 5, and is indented, as the previous chapter requires for subproofs.



```
1. P⇒Q ●
2. Q⇒R ●
3.     P 👍        want R
4.     Q 👍        ⇒E, 1, 3
5.     R 👍        ⇒E, 2, 4
6. P⇒R 👍          ⇒I, 3, 5
```

Figure 6.10: One of the proofs from last chapter's exercises, bubbled and validated in *Lurch*



```
1. P⇒Q ●
2. Q⇒R  subproof context
3.     P 👍        want R
4.     Q 👍        ⇒E, 1, 3
5.     R 👍        ⇒E, 2, 4
6. P⇒R 👍          ⇒I, 3, 5
```

Figure 6.11: The proof from Figure 6.10, but with the cursor inside the subproof, causing *Lurch* to show the subproof's boundaries

Lurch does not actually require you to indent the subproof, because it is capable of detecting the presence of the subproof by the use of the ⇒I rule. Your instructor may still require you to indent subproofs, to prove that you know when you're making assumptions (at the start of subproofs) and discharging them (when the subproof ends), but *Lurch* does not require it. *Lurch* would grade the proof in Figure 6.10 as correct whether or not it included indentation.

Furthermore, recall that the phrase "want R" on line 3 is not actually a part of the proof. It does not get bubbled in *Lurch* in any way, because it's really only an optional note to the human reader (or writer) of the proof, to help them see and remember the proof writer's goals. Thus line 3 has no reason attached to it, yet unlike lines 1 and 2, it has a green thumbs up. Why the difference?

When *Lurch* detects that a subproof has been completed and an assumption discharged, it no longer labels that assumption (the P on line 3 in this case) as a hypothesis of the proof, because it recognizes that it was only a *temporary* hypothesis for a *subproof* that's been completed. While you are working on the subproof, the validation icon next to the P will be yellow, but when you close the subproof it will become green, to indicate that your subproof has been correctly closed and the assumption P correctly discharged.

Furthermore, *Lurch* provides visual feedback to show you where it's detected subproofs in your work. If you place your cursor anywhere in the subproof, you'll see a green (Context) bubble appear around the entire subproof, with a label to indicate that *Lurch* has detected your subproof, as in Figure 6.11. It shows that the subproof begins with the assumption P, and is discharged the moment you conclude the statement $P \Rightarrow R$. This can also be helpful to remind you where you should be indenting lines.

Nearby premises. When writing proofs in *Lurch*, you may find that *Lurch* does not always require you to cite premises, but can auto-locate some of them for you. In fact, it can find any premises that immediately precede the statement you're trying to justify. Check with your instructor as to whether he or she still wants you to cite premises anyway, or if it's acceptable to let *Lurch* detect them for you.

Additional practice. So far this chapter has introduced the method of bubbling statements, reasons, and premises in *Lurch*, but only asked you to do it for one line in one proof. I suggest you start a new numbered list in the same document in which you did the simple, two-line proof from earlier in this chapter, and attempt to recreate the proof in Figures 6.10 and 6.11. That will give you practice with three more sets of reasons and premises, as well as indenting and citing rules other than just R. It will also give you a chance to see a subproof bubble on your own screen.

6.2 Creating your own proofs

In this chapter's practice problems, you'll be asked to create some proofs on your own for the first time. The good news is, now you have *Lurch* to help you, checking each step of your work along the way, and sometimes suggesting how to fix one that's wrong. But 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. I suggest trying some of the practice problems for this chapter, and returning to this list for tips if and when you get stuck.

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 $\mathcal{A} \Rightarrow \mathcal{B}$, plan to use the $\Rightarrow$I rule. This requires starting a subproof in which you assume $\mathcal{A}$. In the subproof, you want to derive $\mathcal{B}$.

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 $\mathcal{A} \vee \mathcal{B}$ as a premise, you know that the $\vee$E rule requires having either $\neg\mathcal{A}$ or $\neg\mathcal{B}$, so you might check to see if you have either of those sentences, or could derive them.

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. Rules for logical equivalence can often make your life easier. If a proof seems impossible, try changing your premises to a different form, or working backwards from a different form of the conclusion.

For example: It is often difficult to prove a disjunction using the basic rules. If you want to show $\mathcal{A} \vee \mathcal{B}$, it is often easier to show $\neg\mathcal{A} \Rightarrow \mathcal{B}$ and then use the MC rule.

Some logical equivalence 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.

6.3 Proof-theoretic concepts

We will use the symbol '$\vdash$' to indicate that a proof is possible. This symbol is called the *turnstile*. When we write $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\} \vdash \mathcal{B}$, it means that it is possible to give a proof of $\mathcal{B}$ with $\mathcal{A}_1$,$\mathcal{A}_2$,... as premises. With just one premise, we leave out the curly braces, so $\mathcal{A} \vdash \mathcal{B}$ means that there is a proof of $\mathcal{B}$ with $\mathcal{A}$ as a premise. Naturally, $\vdash \mathcal{C}$ means that there is a proof of $\mathcal{C}$ that has no premises.

Often, logical proofs are called *derivations*. So $\mathcal{A} \vdash \mathcal{B}$ can be read as '$\mathcal{B}$ is derivable from $\mathcal{A}$.'

A THEOREM is a sentence that is derivable without any premises; i.e., $\mathcal{T}$ is a theorem if and only if $\vdash \mathcal{T}$.

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 $\neg(P \wedge \neg P)$, which shows that $(P \wedge \neg P)$ 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.

Two sentences $\mathcal{A}$ and $\mathcal{B}$ are PROVABLY EQUIVALENT if and only if each can be derived from the other; i.e., $\mathcal{A} \vdash \mathcal{B}$ and $\mathcal{B} \vdash \mathcal{A}$.

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 $\mathcal{A}$ and $\mathcal{B}$ were provably equivalent?)

The set of sentences $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$ is PROVABLY INCONSISTENT if and only if a contradiction is derivable from it; i.e., for some sentence $\mathcal{B}$, $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\} \vdash \mathcal{B}$ and $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\} \vdash \neg\mathcal{B}$.

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*.

6.4 Proofs and truth tables

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

	YES	NO
Is $\mathcal{A}$ a tautology?	prove $\vdash \mathcal{A}$	give a partial truth table in which $\mathcal{A}$ is false
Is $\mathcal{A}$ a contradiction?	prove $\vdash \neg\mathcal{A}$	give a partial truth table in which $\mathcal{A}$ is true
Is $\mathcal{A}$ contingent?	give a partial truth table in which $\mathcal{A}$ is true and another in which $\mathcal{A}$ is false	prove $\vdash \mathcal{A}$ or $\vdash \neg\mathcal{A}$
Are $\mathcal{A}$ and $\mathcal{B}$ equivalent?	prove $\mathcal{A} \vdash \mathcal{B}$ and $\mathcal{B} \vdash \mathcal{A}$	give a partial truth table in which $\mathcal{A}$ and $\mathcal{B}$ have different truth values
Is the set $\mathbb{A}$ consistent?	give a partial truth table in which all the sentences in $\mathbb{A}$ are true	taking the sentences in $\mathbb{A}$, prove $\mathcal{B}$ and $\neg\mathcal{B}$
Is the argument '$\mathcal{P}$, $\therefore$ $\mathcal{C}$' valid?	prove $\mathcal{P} \vdash \mathcal{C}$	give a partial truth table in which $\mathcal{P}$ is true and $\mathcal{C}$ is false

Table 6.1: Sometimes it is easier to show something by providing proofs than it is by providing truth tables. Sometimes it is the other way round. It depends on what you are trying to show.

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 a truth table, which may be very short or very long, depending on the number of different atomic letters in the sentence.

Fortunately, a sentence is a theorem if and only if it is a tautology. (We will discuss this fact further in Chapter 9.) If we provide a proof of $\vdash \mathcal{A}$ and thus show that it is a theorem, it follows that $\mathcal{A}$ is a tautology; i.e., $\models \mathcal{A}$. Similarly, if we find a partial truth table in which $\mathcal{A}$ is false and thus show that it is not a tautology, it follows that $\mathcal{A}$ is not a theorem.

In general, $\mathcal{A} \vdash \mathcal{B}$ if and only if $\mathcal{A} \models \mathcal{B}$. As such, each of the following statements is true; notice that each one connects a concept about truth tables to an equivalent concept about proofs.

- ▷ 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 truth tables, doing whichever is easier for a given task. Table 6.1 summarizes when it is best to give proofs and when it is best to use truth tables.

In this way, proofs and truth tables 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.

Summary of definitions

▷ A sentence $\mathcal{A}$ is a THEOREM if and only if $\vdash \mathcal{A}$.

▷ Two sentences $\mathcal{A}$ and $\mathcal{B}$ are PROVABLY EQUIVALENT if and only if $\mathcal{A} \vdash \mathcal{B}$ and $\mathcal{B} \vdash \mathcal{A}$.

▷ $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$ is PROVABLY INCONSISTENT if and only if, for some sentence $\mathcal{B}$, $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\} \vdash (\mathcal{B} \wedge \neg\mathcal{B})$.

Quiz Yourself

- If I have a proof of a wff of SL, does that make the wff a tautology?
- For some wff $\mathcal{A}$ of SL, if $\vdash \neg\mathcal{A}$ then what do we know about the truth table for $\mathcal{A}$?
- To which section of this chapter should you turn for strategic advice when you're stuck on a proof?

Practice Exercises

It is not required that you do any of the following work in *Lurch*, but doing so provides helpful feedback on correctness. I recommend taking advantage of that feedback.

Part A For each of the proofs you typed in *Lurch* as part of the exercises in the previous chapter, add bubbles for reasons and premises, and ensure that *Lurch* validates each step in each proof.

Part B Give a proof for each argument in SL.

1. $K \wedge L$, $\therefore K \Leftrightarrow L$
2. $A \Rightarrow (B \Rightarrow C)$, $\therefore (A \wedge B) \Rightarrow C$
3. $P \wedge (Q \vee R)$, $P \Rightarrow \neg R$, $\therefore Q \vee E$
4. $(C \wedge D) \vee E$, $\therefore E \vee D$
5. $\neg F \Rightarrow G$, $F \Rightarrow H$, $\therefore G \vee H$
6. $(X \wedge Y) \vee (X \wedge Z)$, $\neg(X \wedge D)$, $D \vee M$ $\therefore M$

Part C Give a proof for each argument in SL.

1. $Q \Rightarrow (Q \wedge \neg Q)$, $\therefore \neg Q$
2. $J \Rightarrow \neg J$, $\therefore \neg J$
3. $E \vee F$, $F \vee G$, $\neg F$, $\therefore E \wedge G$
4. $A \Leftrightarrow B$, $B \Leftrightarrow C$, $\therefore A \Leftrightarrow C$
5. $M \vee (N \Rightarrow M)$, $\therefore \neg M \Rightarrow \neg N$
6. $S \Leftrightarrow T$, $\therefore S \Leftrightarrow (T \vee S)$
7. $(M \vee N) \wedge (O \vee P)$, $N \Rightarrow P$, $\neg P$, $\therefore M \wedge O$
8. $(Z \wedge K) \vee (K \wedge M)$, $K \Rightarrow D$, $\therefore D$

Part D Show that each of the following sentences is a theorem in SL.

1. $O \Rightarrow O$

2. $N \vee \neg N$
3. $\neg(P \wedge \neg P)$
4. $\neg(A \Rightarrow \neg C) \Rightarrow (A \Rightarrow C)$
5. $J \Leftrightarrow (J \vee (L \wedge \neg L))$

Part E Show that each of the following pairs of sentences are provably equivalent in SL.

1. $\neg\neg\neg\neg G$, G
2. $T \Rightarrow S$, $\neg S \Rightarrow \neg T$
3. $R \Leftrightarrow E$, $E \Leftrightarrow R$
4. $\neg G \Leftrightarrow H$, $\neg(G \Leftrightarrow H)$
5. $U \Rightarrow I$, $\neg(U \wedge \neg I)$

Part F Provide proofs to show each of the following.

1. $M \wedge (\neg N \Rightarrow \neg M) \vdash (N \wedge M) \vee \neg M$
2. $\{C \Rightarrow (E \wedge G), \neg C \Rightarrow G\} \vdash G$
3. $\{(Z \wedge K) \Leftrightarrow (Y \wedge M), D \wedge (D \Rightarrow M)\} \vdash Y \Rightarrow Z$
4. $\{(W \vee X) \vee (Y \vee Z), X \Rightarrow Y, \neg Z\} \vdash W \vee Y$

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 or logical equivalences.

1. Show that MT is a legitimate derived rule. Using only the basic rules, prove the following: $\mathcal{A} \Rightarrow \mathcal{B}$, $\neg\mathcal{B}$, $\therefore \neg\mathcal{A}$
2. Show that Comm is a legitimate rule for the biconditional. Using only the basic rules, prove that $\mathcal{A} \Leftrightarrow \mathcal{B}$ and $\mathcal{B} \Leftrightarrow \mathcal{A}$ are equivalent.
3. Using only the basic rules, prove the following instance of DeMorgan's Laws: $(\neg A \wedge \neg B)$, $\therefore \neg(A \vee B)$
4. Show that $\Leftrightarrow$ex is a legitimate derived rule. Using only the basic rules, prove that $D \Leftrightarrow E$ and $(D \Rightarrow E) \wedge (E \Rightarrow D)$ are equivalent.

Part H

1. If you know that $\mathcal{A} \vdash \mathcal{B}$, what can you say about $(\mathcal{A} \wedge \mathcal{C}) \vdash \mathcal{B}$? Explain your answer.
2. If you know that $\mathcal{A} \vdash \mathcal{B}$, what can you say about $(\mathcal{A} \vee \mathcal{C}) \vdash \mathcal{B}$? Explain your answer.

Chapter 7

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.

7.1 From sentences to predicates

Consider the following argument, which is obviously 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.
> This is a logic class, so 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 could not replace E with $\neg N$. Why not? Because $\neg N$ 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.

Once we have separate sentence letters for N and E, however, we have erased any connection between the two. They are just two atomic sentences which might be true or false independently. In English, it could never 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.

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. Fortunately, the quantifier structure is not what makes this argument valid. As such, we can safely ignore it. To see this, we translate the argument to SL:

$$\begin{array}{rl} & L \Rightarrow (N \vee E) \\ & E \Rightarrow B \\ & L \\ \therefore & \neg B \Rightarrow N \end{array}$$

This is a valid argument in SL. (You can do a truth table or a proof to check 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:

$$\begin{array}{rl} & L \\ & A \\ \therefore & F \end{array}$$

This is *invalid* in SL. (Again, you can confirm this with a truth table.) There is something very wrong here, because this is clearly a *valid* argument in English. The symbolization in SL leaves out all the important structure. Once again, the translation to SL overlooks quantifier structure: The sentence 'All logicians wear funny hats' is about both logicians and hat-wearing. By not translating this structure, we lose the connection between Willard's being a logician and Willard's wearing a hat.

The point is this: Some arguments with quantifier structure, like the first example, can be captured in SL even though SL ignores the quantifier structure. Other arguments are completely botched in SL, like the second example. Notice that the problem is not that we have made a mistake while symbolizing the second argument. These are the best symbolizations we can give for these arguments *in SL*.

Generally, if an argument containing quantifiers comes out *valid in SL*, then the English language argument is valid. If it comes out *invalid in SL*, then we cannot say the English language argument is invalid. The argument might be valid because of quantifier structure which the natural language argument has and which the argument in SL lacks.

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 logical language. We will call this language quantified logic, QL.

7.2 Building blocks of QL

Just as sentences were the basic unit of sentential logic, predicates will be the basic unit of quantified logic. A predicate is an expression like 'is a dog.' This is not a sentence on its own. It is neither true nor false. In order to be true or false, we need to specify something: Who or what is it that is a dog?

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 lower-case words. For instance, we might let *dog* stand for '_____ is a dog.' We will use upper-case letters as the names of specific things. For instance, we might let B stand for Bertie. The expression $dog(B)$ will be a sentence in QL. It 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, '$\exists$' will mean 'There is some _____.' So to say that there is a dog, we can write $\exists x, dog(x)$; that is: There is some x such that x is a dog.

That will come later. We start by defining singular terms and predicates.

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 there are a great many dogs. The phrase 'Philip's dog Bertie' is a singular term, because it refers to a specific little terrier.

A PROPER NAME is a singular term that picks out an individual without describing it. 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 other 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 'Philip's dog Bertie' 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 can use proper names and definite descriptions to pick out the same thing. The proper name 'Mount Rainier' names the location picked out by the definite description 'the highest peak in Washington state.' The expressions refer to the same place in different ways. You learn nothing from my saying that I am going to Mount Rainier, unless you already know some geography. You could guess that it is a mountain, perhaps, but even this is not a sure thing; for all you know it might be a college, like Mount Holyoke. Yet if I were to say that I was going to the highest peak in Washington state, you would know immediately that I was going to a mountain in Washington state.

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 first author of this text and not the other P.D. Magnus. 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 capitalized letters. So A, B, C, ..., Y, and Z are all terms in QL. Singular terms are called CONSTANTS because they pick out specific individuals.

Note that a, b, c, ..., y, and z are not constants in QL. They will be VARIABLES, terms which do not stand for any specific thing. We will need them when we introduce quantifiers.

Predicates

The simplest predicates are properties of individuals. They are things you can say about an object. '_____ is a dog' and '_____ is a member of Monty Python' are both predicates. In translating English sentences, the term will not always come at the beginning of the sentence: 'A piano fell on _____' is also a predicate. 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 _____', and '_____ 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, 'Vinnie borrowed the family car from Nunzio.' By removing a singular term, we can recognize this sentence as using any of three different monadic predicates:

_____ borrowed the family car from Nunzio.
Vinnie borrowed _____ from Nunzio.
Vinnie borrowed the family car from _____.

By removing two singular terms, we can recognize three different dyadic predicates:

Vinnie borrowed _____ from _____.
_____ borrowed the family car from _____.
_____ borrowed _____ from Nunzio.

By removing all three singular terms, we can recognize one three-place or triadic predicate:

_____ borrowed _____ from _____.

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 the only thing that we will discuss being borrowed is the family car, then the generality of the three-place predicate is unnecessary. If the only borrowing we need to symbolize is different people borrowing the family car from Nunzio, then 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.

In QL, we symbolize predicates with lower-case words at least two letters long. 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:

$angry(x)$:	x is angry.
$happy(x)$:	x is happy.
$tall(x,y)$:	x is as tall or taller than y.
$tough(x,y)$:	x is as tough or tougher than y.
$btwn(x,y,z)$:	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: $angry(D)$. The 'x' in the key entry '$angry(x)$' is just a placeholder; we can replace it with other terms when translating.

Sentence 2 can be paraphrased as, 'If $angry(D)$, then $angry(G)$ and $angry(M)$.' QL has all the truth-functional connectives of SL, so we translate this as $angry(D) \Rightarrow (angry(G) \wedge angry(M))$.

Sentence 3 can be translated as $tall(M, G) \wedge tough(M, G)$.

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 $short(x, y)$ to mean 'x is shorter than y.' However, this would ignore the logical connection between 'shorter' and 'taller.' Considered only as predicates of QL, there is no connection between *short* and *tall*. 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 $\neg tall(D, G)$.

Sentence 5 requires that we pay careful attention to the order of terms in the key. It becomes $btwn(D, G, M)$.

Quiz Yourself

- Why are we learning a second formal language when we already know one?
- Is "the first winner of the Stanley Cup" a proper name? A definite description? A singular term?
- What is a one-place predicate?

7.3 Quantifiers

We are now ready to introduce quantifiers. Consider these 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 $happy(D) \wedge happy(G) \wedge happy(M)$. 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 '$\forall$' symbol. This is called the UNIVERSAL QUANTIFIER.

A quantifier must always be followed by a variable, then a comma, and then a formula that includes that variable. We can translate sentence 6 as $\forall x, happy(x)$. Paraphrased in English, this means 'For all x, x is happy.' We call $\forall x$ an *x-quantifier*. The formula that follows the comma 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 $\forall x, happy(x)$, the scope of the universal quantifier is $happy(x)$.

Sentence 7 can be paraphrased as, 'For all x, x is at least as tough as Donald.' This translates as $\forall x, tough(x, D)$.

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 $\forall x, happy(x)$ means exactly the same thing as $\forall y, happy(y)$, $\forall z, happy(z)$, and $\forall a, happy(a)$.

To translate sentence 8, we introduce another new symbol: the EXISTENTIAL QUANTIFIER, $\exists$. Like the universal quantifier, the existential quantifier requires a variable. Sentence 8 can be translated as $\exists x, angry(x)$. 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 $\exists t, angry(t)$.

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: $\neg\exists x, angry(x)$. 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: $\forall x, \neg angry(x)$. 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, $\forall x, \mathcal{A}$ is logically equivalent to $\neg\exists x, \neg\mathcal{A}$. 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 $\exists x, \neg happy(x)$. Equivalently, we could write $\neg\forall x, happy(x)$.

Sentence 11 is most naturally translated as $\neg\forall x, happy(x)$. This is logically equivalent to sentence 10 and so could also be translated as $\exists x, \neg happy(x)$.

Although we have two quantifiers in QL, we could have an equivalent formal language with only one quantifier. We could proceed with only the universal quantifier, for instance, and treat the existential quantifier as a notational convention. We could write '$\exists x$' knowing that this is just shorthand for '$\neg\forall x, \neg$.' There is a choice between making logic formally simple and making it expressively simple. With QL, we opt for expressive simplicity. Both $\forall$ and $\exists$ will be symbols of QL.

Universe of Discourse

Given the symbolization key we have been using, $\forall x, happy(x)$ 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.

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
$paris(x)$**:** x is in Paris.

The sentence $\forall x, paris(x)$ 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 $\forall x, paris(x)$ just means that the Eiffel Tower is in Paris.

Non-referring terms

In QL, each constant must pick out exactly one member of the UD. A constant cannot refer to more than one thing— it is a *singular* term. Each constant must still pick out *something.* This is connected to a classic philosophical problem: the so-called problem of non-referring terms.

Medieval philosophers typically used sentences about the *chimera* to exemplify this problem. Chimera is a mythological creature; it does not really exist. Consider these two sentences:

12. Chimera is angry.
13. Chimera is not angry.

It is tempting just to define a constant to mean 'chimera.' The symbolization key would look like this:

UD: creatures on Earth
$angry(x)$**:** x is angry.
C**:** chimera

We could then translate sentence 12 as $angry(C)$ and sentence 13 as $\neg angry(C)$.

Problems will arise when we ask whether these sentences are true or false. One option is to say that sentence 12 is not true, because there is no chimera. If sentence 12 is false because it talks about a non-existent thing, then sentence 13 is false for the same reason. Yet this would mean that $angry(C)$ and $\neg angry(C)$ would both be false. Given the truth conditions for negation, this cannot be the case.

Since we cannot say that they are both false, what should we do? Another option is to say that sentence 12 is *meaningless* because it talks about a non-existent thing. So $angry(C)$ would be a meaningful expression in QL for some interpretations but not for others. Yet this would make our formal language hostage to particular interpretations. Since we are interested in logical form, we want to consider the logical force of a sentence like $angry(C)$ apart from any particular interpretation. If $angry(C)$ were sometimes meaningful and sometimes meaningless, we could not do that.

This is the *problem of non-referring terms*, and we will return to it later (see p. 95). The important point for now is that each constant of QL *must* refer to something in the UD, although the UD can be any set of things that we like. If we want to symbolize arguments about mythological creatures, then we must define a UD that includes them. This option is important if we want to consider the logic of stories. We can translate a sentence like 'Sherlock Holmes lived at 221B Baker Street' by including fictional characters like Sherlock Holmes in our UD.

Quiz Yourself

- What are the two quantifiers in QL?
- Why must we specify a universe of discourse?

7.4 Translating to QL

We now have all of the pieces of QL. Translating more complicated sentences will only be a matter of knowing the right way to combine predicates, constants, quantifiers, and connectives. Consider these sentences:

14. Every coin in my pocket is a quarter.
15. Some coin on the table is a dime.
16. Not all the coins on the table are dimes.
17. 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
$pocket(x)$**:**	x is in my pocket.
$table(x)$**:**	x is on the table.
$quarter(x)$**:**	x is a quarter.
$dime(x)$**:**	x is a dime.

Sentence 14 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 14 means that, for any coin, *if* that coin is in my pocket *then* it is a quarter. So we can translate it as $\forall x, (pocket(x) \Rightarrow quarter(x))$.

Since sentence 14 is about coins that are both in my pocket *and* that are quarters, it might be tempting to translate it using a conjunction. However, the sentence $\forall x, (pocket(x) \wedge quarter(x))$ would mean that everything in the UD is both in my pocket and a quarter: All the coins that exist are quarters in my pocket. This would be a crazy thing to say, and it means something very different than sentence 14.

Sentence 15 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 $\exists x, (\mathit{table}(x) \wedge \mathit{dime}(x))$.

Notice that we needed to use a conditional with the universal quantifier, but we used a conjunction with the existential quantifier. What would it mean to write $\exists x, (\mathit{table}(x) \Rightarrow \mathit{dime}(x))$? Probably not what you think. It means that there is some member of the UD which would satisfy the conditional subformula. In SL, $\mathcal{A} \Rightarrow \mathcal{B}$ is logically equivalent to $\neg\mathcal{A} \vee \mathcal{B}$, and this will also hold in QL. So $\exists x, (\mathit{table}(x) \Rightarrow \mathit{dime}(x))$ is true if there is some coin that is *either* not on the table *or* is a dime. Of course there is a coin that is not the table— there are coins lots of other places. So $\exists x, (\mathit{table}(x) \Rightarrow \mathit{dime}(x))$ is trivially true. 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. As a general rule, do not put conditionals in the scope of existential quantifiers unless you are sure that you need one.

Sentence 16 can be paraphrased as, 'It is not the case that every coin on the table is a dime.' So we can translate it as $\neg\forall x, (\mathit{table}(x) \Rightarrow \mathit{dime}(x))$. You might look at sentence 16 and paraphrase it instead as, 'Some coin on the table is not a dime.' You would then translate it as $\exists x, (\mathit{table}(x) \wedge \neg\mathit{dime}(x))$. Although it is probably not obvious, these two translations are logically equivalent. (This is due to the logical equivalence between $\neg\forall x, \mathcal{A}$ and $\exists x, \neg\mathcal{A}$, along with the equivalence between $\neg(\mathcal{A} \Rightarrow \mathcal{B})$ and $\mathcal{A} \wedge \neg\mathcal{B}$.)

Sentence 17 can be paraphrased as, 'It is not the case that there is some dime in my pocket.' This can be translated as $\neg\exists x, (\mathit{pocket}(x) \wedge \mathit{dime}(x))$. It might also be paraphrased as, 'Everything in my pocket is a non-dime,' and then could be translated as $\forall x, (\mathit{pocket}(x) \Rightarrow \neg\mathit{dime}(x))$. Again the two translations are logically equivalent. Both are correct translations of sentence 17.

We can now translate the argument from p. 76, the one that motivated the need for quantifiers:

Willard is a logician. All logicians wear funny hats.
∴ Willard wears a funny hat.

UD:	people
$\mathit{log}(x)$**:**	x is a logician.
$\mathit{funny}(x)$**:**	x wears a funny hat.
W**:**	Willard

Translating, we get:

$$\begin{array}{ll} & \mathit{log}(W) \\ & \forall x, (\mathit{log}(x) \Rightarrow \mathit{funny}(x)) \\ \therefore & \mathit{funny}(W) \end{array}$$

This captures the structure that was left out of the SL translation of this argument, and this is a valid argument in QL.

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:

18. Every monkey knows sign language.

19. Some monkey knows sign language.

It is possible to write the symbolization key for these sentences in this way:

UD:	animals
$mon(x)$**:**	x is a monkey.
$sign(x)$**:**	x knows sign language.

Sentence 18 can now be translated as $\forall x, (mon(x) \Rightarrow sign(x))$.

Sentence 19 becomes $\exists x, (mon(x) \wedge sign(x))$.

It is tempting to say that sentence 18 entails sentence 19; that is: if every monkey knows sign language, then it must be that some monkey knows sign language. This is a valid inference in Aristotelean logic, but the entailment does not hold in QL. It is possible for the sentence $\forall x, (mon(x) \Rightarrow sign(x))$ to be true even though the sentence $\exists x, (mon(x) \wedge sign(x))$ is false.

How can this be? The answer comes from considering whether these sentences would be true or false *if there were no monkeys.*

We have defined $\forall$ and $\exists$ in such a way that $\forall x, \mathcal{A}$ is equivalent to $\neg\exists x, \neg\mathcal{A}$. As such, the universal quantifier doesn't involve the existence of anything— only non-existence. If sentence 18 is true, then there are *no* monkeys who don't know sign language. If there were no monkeys, then $\forall x, (mon(x) \Rightarrow sign(x))$ would be true and $\exists x, (mon(x) \wedge sign(x))$ would be false.

We allow empty predicates because 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.

What happens if we add an empty predicate ref to the interpretation above? For example, we might define $ref(x)$ to mean 'x is a refrigerator.' Now the sentence $\forall x, (ref(x) \Rightarrow mon(x))$ 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 $\forall x, (ref(x) \Rightarrow mon(x))$ means that any member of the UD that 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 ref would not be empty and the sentence $\forall x, (ref(x) \Rightarrow mon(x))$ would be false.

- ▷ 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.

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 $funny(x)$ means 'x wears a funny hat' or 'x tells funny jokes.' The meaning of sentences in QL also depends on the UD.

Let $rose(x)$ mean 'x is a rose,' let $thorn(x)$ mean 'x has a thorn,' and consider this sentence:

20. Every rose has a thorn.

It is tempting to say that sentence 20 should be translated as $\forall x, (rose(x) \Rightarrow thorn(x))$. If the UD contains all roses, that would be correct. Yet if the UD is merely *things on my kitchen table*, then $\forall x, (rose(x) \Rightarrow thorn(x))$ 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 in the UD all roses in order to translate sentence 20. We have two options. First, we can restrict the UD to include all roses but *only* roses. Then sentence 20 becomes $\forall x, thorn(x)$. 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 20 must be $\forall x, (rose(x) \Rightarrow thorn(x))$.

If we wanted the universal quantifier to mean *every* thing, without restriction, then we might try to specify a UD that contains everything. This would lead to problems. 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 20 and, with the same symbolization key, translate these sentences:

21. Esmerelda has a rose in her hair.
22. Everyone is cross with Esmerelda.

We need a UD that includes roses (so that we can symbolize sentence 20) and a UD that includes people (so we can translate sentences 21–22.) Here is a suitable key:

UD:	people and plants
$per(x)$**:**	x is a person.
$rose(x)$**:**	x is a rose.
$thorn(x)$**:**	x has a thorn.
$cross(x, y)$**:**	x is cross with y.
$hair(x, y)$**:**	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 21 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: $\exists x, (rose(x) \wedge hair(E, x))$.

It is tempting to translate sentence 22 as $\forall x, cross(x, E)$. 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 22 does not mean that.

'Everyone' means every person, not every member of the UD. So we can paraphrase sentence 22 as, 'Every person is cross with Esmerelda.' We know how to translate such sentences: $\forall x, (per(x) \Rightarrow cross(x, E))$

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.'

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
$guitar(x)$**:**	x can play guitar.
$rock(x)$**:**	x is a rock star.
L**:**	Lemmy

Now consider these sentences:

23. If Lemmy can play guitar, then he is a rock star.
24. If a person can play guitar, then he is a rock star.

Sentence 23 and sentence 24 have the same consequent ('. . . he is 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 23 can be paraphrased as, 'If Lemmy can play guitar, then *Lemmy* is a rockstar.' This can obviously be translated as $guitar(L) \Rightarrow rock(L)$.

Sentence 24 must be paraphrased differently: 'If a person can play guitar, then *that person* is a rock star.' This sentence 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 rockstar.' Now this can be translated as $\forall x, (guitar(x) \Rightarrow rock(x))$. This is the same as, 'Everyone who can play guitar is a rock star.'

Consider these further sentences:

25. If anyone can play guitar, then Lemmy can.
26. If anyone can play guitar, then he or she is a rock star.

These two sentences have the same antecedent ('If anyone can play guitar. . .'), but they have different logical structures.

Sentence 25 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: $\exists x, guitar(x) \Rightarrow guitar(L)$.

Sentence 26 can be paraphrased, 'For any person, if that person can play guitar, then that person 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 $\forall x, (guitar(x) \Rightarrow rock(x))$.

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 25) and sometimes for a universal quantifier (as in sentence 26). 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.'

Quantifiers and scope

In the sentence $\exists x, guitar(x) \Rightarrow guitar(L)$, the scope of the existential quantifier is the expression $guitar(x)$. Would it matter if the scope of the quantifier were the whole sentence? That is, does the sentence $\exists x, (guitar(x) \Rightarrow guitar(L))$ mean something different?

With the key given above, $\exists x, guitar(x) \Rightarrow guitar(L)$ means that if there is some guitarist, then Lemmy is a guitarist. But $\exists x, (guitar(x) \Rightarrow guitar(L))$ 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 if the antecedent is false. Let the constant P denote the author of this book, someone who is certainly not a guitarist. The sentence $guitar(P) \Rightarrow guitar(L)$ is true because $guitar(P)$ is false. Since someone (namely P) satisfies the sentence, then $\exists x, (guitar(x) \Rightarrow guitar(L))$ 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: $\exists x, guitar(x) \Rightarrow guitar(L)$ means the same thing as $\forall x, (guitar(x) \Rightarrow guitar(L))$, and $\exists x, (guitar(x) \Rightarrow guitar(L))$ means the same thing as $\forall x, guitar(x) \Rightarrow guitar(L)$.

This oddity does not arise with other connectives or if the variable is in the consequent of the conditional. For example, $\exists x, guitar(x) \wedge guitar(L)$ means the same thing as $\exists x, (guitar(x) \wedge guitar(L))$, and $guitar(L) \Rightarrow \exists x, guitar(x)$ means the same things as $\exists x, (guitar(L) \Rightarrow guitar(x))$.

Ambiguous predicates

Suppose we just want to translate this sentence:

27. Adina is a skilled surgeon.

Let the UD be people, let $ss(x)$ mean 'x is a skilled surgeon', and let A mean Adina. Sentence 27 is simply $ss(A)$.

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
$\mathit{greedy}(x)$**:**	x is greedy.
$\mathit{hire}(x)$**:**	The hospital will hire x.
$\mathit{sur}(x)$**:**	x is a surgeon.
$\mathit{skill}(x)$**:**	x is skilled.
B**:**	Billy

Now the argument can be translated in this way:

$$\begin{array}{ll} & \forall x, \big(\neg(\mathit{sur}(x) \wedge \mathit{skill}(x)) \Rightarrow \neg \mathit{hire}(x)\big) \\ & \forall x, (\mathit{sur}(x) \Rightarrow \mathit{greedy}(x)) \\ & \mathit{sur}(B) \wedge \neg \mathit{skill}(B) \\ \therefore & \mathit{greedy}(B) \wedge \neg \mathit{hire}(B) \end{array}$$

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 $\mathit{tennis}(x)$ mean 'x is a tennis player') and a constant (let C mean Carol). Then the argument becomes:

$$\begin{array}{ll} & (\mathit{sur}(C) \wedge \mathit{skill}(C)) \wedge \mathit{tennis}(C) \\ \therefore & \mathit{tennis}(C) \wedge \mathit{skill}(C) \end{array}$$

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 $\mathit{ss}(x)$ mean 'x is skilled as a surgeon' and $\mathit{st}(x)$ mean 'x is skilled as a tennis player,' then we can symbolized the argument in this way:

$$\begin{array}{ll} & (\mathit{sur}(C) \wedge \mathit{ss}(C)) \wedge \mathit{tennis}(C) \\ \therefore & \mathit{tennis}(C) \wedge \mathit{st}(C) \end{array}$$

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 27 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.'

Multiple quantifiers

Consider the following symbolization key and the sentences that follow it:

UD:	People and dogs
$dog(x)$**:**	x is a dog.
$friend(x, y)$**:**	x is a friend of y.
$owns(x, y)$**:**	x owns y.
F**:**	Fifi
G**:**	Gerald

28. Fifi is a dog.
29. Gerald is a dog owner.
30. Someone is a dog owner.
31. All of Gerald's friends are dog owners.
32. Every dog owner is the friend of a dog owner.

Sentence 28 is easy: $dog(F)$.

Sentence 29 can be paraphrased as, 'There is a dog that Gerald owns.' This can be translated as $\exists x, (dog(x) \wedge owns(G, x))$.

Sentence 30 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 29, except that it is about y rather than being about Gerald. So we can translate sentence 30 as $\exists y, \exists x, (dog(x) \wedge owns(y, x))$.

Sentence 31 can be paraphrased as, 'Every friend of Gerald is a dog owner.' Translating part of this sentence, we get $\forall x, (friend(x, G) \Rightarrow$ 'x is a dog owner'). Again, it is important to recognize that 'x is a dog owner' is structurally just like sentence 29. Since we already have an x-quantifier, we will need a different variable for the existential quantifier. Any other variable will do. Using d for dog, sentence 31 can be translated as $\forall x, \big(friend(x, G) \Rightarrow \exists d, (dog(d) \wedge owns(x, d))\big)$.

Sentence 32 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

$$\forall x, \big(x \text{ is a dog owner} \Rightarrow \exists y(y \text{ is a dog owner} \wedge friend(x, y))\big).$$

Completing the translation, sentence 32 becomes

$$\forall x, \big(\exists d, (dog(d) \wedge owns(x, d)) \Rightarrow \exists y, \big(\exists d, (dog(d) \wedge owns(y, d)) \wedge friend(x, y)\big)\big).$$

Consider this symbolization key and these sentences:

UD:	people
$likes(x, y)$**:**	x likes y.
I**:**	Imre
K**:**	Karl

33. Imre likes everyone that Karl likes.
34. There is someone who likes everyone who likes everyone that he likes.

Sentence 33 can be partially translated as $\forall x$, (Karl likes $x \Rightarrow$ Imre likes x). This becomes $\forall x, (likes(K, x) \Rightarrow likes(I, x))$.

Sentence 34 is almost a tongue-twister. 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:

$$\exists x, \forall y, (y \text{ likes everyone that } x \text{ likes} \Rightarrow x \text{ likes } y).$$

The antecedent of the conditional is structurally just like sentence 33, with y and x in place of Imre and Karl. So sentence 34 can be completely translated in this way

$$\exists x, \forall y, \big(\forall z, (likes(x, z) \Rightarrow likes(y, z)) \Rightarrow likes(x, y)\big)$$

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 phrases.

Quiz Yourself

- Which logical connective commonly belongs immediately inside a universal quantifier?
- Which logical connective commonly belongs immediately inside an existential quantifier?
- Why is the phrase “skilled surgeon” an ambiguous predicate?
- Give an example of an English sentence that requires more than one quantifier when translated into QL, and explain why it needs more than one.

7.5 Sentences of QL

In this section, we provide a formal definition for a *well-formed formula* (wff) and *sentence* of QL.

Expressions

There are six kinds of symbols in QL:

predicates	$aa, ab, ac, \ldots, zx, zy, zz, aaa, aab, \ldots,$ $angry, happy, dog, squeamish, \ldots$
constants	$A, B, C, \ldots, X, Y, Z$
variables	$a, b, c, \ldots, x, y, z$
connectives	$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
parentheses	(,)
quantifiers	$\forall, \exists$
comma	,

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.

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 parentheses surrounding n terms separated by commas.

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 for 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 $\mathscr{A}$ is a wff, then $\forall x, \mathscr{A}$ and $\exists x, \mathscr{A}$ are wffs. However, this would allow for bizarre sentences like $\forall x, \exists x, dog(x)$ and $\forall x, dog(W)$. What could these possibly mean? We could adopt some interpretation of such sentences, but instead we will write the definition of a wff so that such abominations do not even count as well-formed.

In order for $\forall x, \mathscr{A}$ to be a wff, $\mathscr{A}$ must contain the variable x and must not already contain an x-quantifier. $\forall x, dog(W)$ will not count as a wff because 'x' does not occur in $dog(W)$, and $\forall x, \exists x, dog(x)$ will not count as a wff because $\exists x, dog(x)$ contains an x-quantifier.

1. Every atomic formula is a wff.
2. If $\mathscr{A}$ is a wff, then $\neg\mathscr{A}$ is a wff.
3. If $\mathscr{A}$ and $\mathscr{B}$ are wffs, then $(\mathscr{A} \wedge \mathscr{B})$, is a wff.
4. If $\mathscr{A}$ and $\mathscr{B}$ are wffs, then $(\mathscr{A} \vee \mathscr{B})$ is a wff.
5. If $\mathscr{A}$ and $\mathscr{B}$ are wffs, then $(\mathscr{A} \Rightarrow \mathscr{B})$ is a wff.
6. If $\mathscr{A}$ and $\mathscr{B}$ are wffs, then $(\mathscr{A} \Leftrightarrow \mathscr{B})$ is a wff.
7. If $\mathscr{A}$ is a wff, χ is a variable, $\mathscr{A}$ contains at least one occurrence of χ, and $\mathscr{A}$ contains no χ-quantifiers, then $\forall\chi, \mathscr{A}$ is a wff.
8. If $\mathscr{A}$ is a wff, χ is a variable, $\mathscr{A}$ contains at least one occurrence of χ, and $\mathscr{A}$ contains no χ-quantifiers, then $\exists\chi, \mathscr{A}$ 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 $\forall x, angry(x)$ is a wff, but so are $\forall y, angry(y)$, $\forall z, angry(z)$, $\forall m, angry(m)$, and $\forall g, angry(g)$.

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.

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
$loves(x, y)$**:**	x loves y
B**:**	Boris

Consider the expression $loves(z, z)$. It is an atomic forumula: a two-place predicate followed by two terms. All atomic formula are wffs, so $loves(z, z)$ is a wff. Does it mean anything? You might think that it means that z loves himself, in the same way that $loves(B, B)$ means that Boris loves himself. Yet z is a variable; it does not name some person the way a constant would. The wff $loves(z, z)$ does not tell us how to interpret z. Does it mean everyone? Anyone? Someone? If we had a z-quantifier, it would tell us how to interpret z. For instance, $\exists z, loves(z, z)$ would mean that someone loves themself.

Some formal languages treat a wff like $loves(z, z)$ as implicitly having a universal quantifier in front. We will not do this for QL. If you mean to say that everyone loves themself, then you need to write the quantifier: $\forall z, loves(z, z)$

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 $\mathcal{X}$ that is within the scope of an $\mathcal{X}$-quantifier. A FREE VARIABLE is an occurance of a variable that is not bound.

For example, consider the wff $\forall x, (egg(x) \vee dog(y)) \Rightarrow \exists z, (egg(x) \Rightarrow loves(z, x))$. The scope of the universal quantifier $\forall x$ is $(egg(x) \vee dog(y))$, so the first x is bound by the universal quantifier but the second and third x's are free. There is no y-quantifier, so the y is free. The scope of the existential quantifier $\exists z$ is $(egg(x) \Rightarrow loves(z, x))$, so both occurrences of z are bound by it.

We define a SENTENCE of QL as a wff of QL that contains no free variables.

Notational conventions

We will adopt the same notational conventions that we did for SL (p. 34). First, we may leave off the outermost parentheses of a formula. Second, we will leave out parentheses between each pair of conjuncts when writing long series of conjunctions. Third, we will leave out parentheses between each pair of disjuncts when writing long series of disjunctions.

Substitution instance

If $\mathcal{A}$ is a wff, C a constant, and x a variable, then $\mathcal{A}[x = C]$ is the wff made by replacing each occurance of x in $\mathcal{A}$ with C. This is called a SUBSTITUTION INSTANCE of $\forall x, \mathcal{A}$ and $\exists x, \mathcal{A}$; C is called the INSTANTIATING CONSTANT.

For example: $angry(A) \Rightarrow bad(A)$, $angry(F) \Rightarrow bad(F)$, and $angry(K) \Rightarrow bad(K)$ are all substitution instances of $\forall x, (angry(x) \Rightarrow bad(x))$; the instantiating constants are A, F, and K, respectively. Similarly, $related(A, J)$, $related(D, J)$, and $related(J, J)$ are substitution instances of $\exists z, related(z, J)$; the instantiating constants are A, D, and J, respectively.

This definition will be useful later, when we define truth and derivability in QL. If $\forall x, pred(x)$ is true, then every substitution instance $pred(A)$, $pred(B)$, $pred(C)$, $\ldots$ is true. To put the point informally, if everything makes $pred$ true, then A does, B does, C does, and so on. Conversely, if some substitution instance of $\exists x, pred(x)$ such as $pred(A)$ is true, then $\exists x, pred(x)$ must be true. Informally, if some specific A makes $pred$ true, then there is something that makes $pred$ true.

7.6 Identity

Consider this sentence:

35. Pavel owes money to everyone else.

Let the UD be people; this will allow us to translate ‘everyone’ as a universal quantifier. Let $owes(x, y)$ mean ‘x owes money to y’, and let P mean Pavel. Now we can symbolize sentence 35 as $\forall x, owes(P, x)$. Unfortunately, this translation has some odd consequences. It says that Pavel owes money to every member of the UD, including Pavel; it entails that Pavel owes money to himself. However, sentence 35 does not say that Pavel owes money to himself; he owes money to everyone *else*. This is a problem, because $\forall x, owes(P, x)$ is the best translation we can give of this sentence into QL.

The solution is to add another symbol to QL. The symbol ‘=’ is a two-place predicate. Since it has a special logical meaning, we write it a bit differently: For two terms t_1 and t_2, $t_1 = t_2$ is an atomic formula.

The predicate $x = y$ means ‘x is identical to y.’ This does not mean merely that x and y are indistinguishable or that all of the same predicates are true of them. Rather, it means that x and y are the very same thing.

When we write $x \neq y$, we mean that x and y are not identical. There is no reason to introduce this as an additional predicate. Instead, $x \neq y$ is an abbreviation of $\neg(x = y)$.

Now suppose we want to symbolize this sentence:

36. Pavel is Mister Checkov.

Let the constant C mean Mister Checkov. Sentence 36 can be symbolized as $P = C$. This means that the constants P and C both refer to the same guy.

This is all well and good, but how does it help with sentence 35? That sentence can be paraphrased as, ‘Everyone who is not Pavel is owed money by Pavel.’ This is a sentence structure we already know how to symbolize: ‘For all x, if x is not Pavel, then x is owed money by Pavel.’ In QL with identity, this becomes $\forall x, (x \neq P \Rightarrow owes(P, x))$.

In addition to sentences that use the word ‘else’, identity will be helpful when symbolizing some sentences that contain the words ‘besides’ and ‘only.’ Consider these examples:

37. No one besides Pavel owes money to Hikaru.
38. Only Pavel owes Hikaru money.

We add the constant H, which means Hikaru.

Sentence 37 can be paraphrased as, ‘No one who is not Pavel owes money to Hikaru.’ This can be translated as $\neg\exists x, (x \neq P \wedge owes(x, H))$.

Sentence 38 can be paraphrased as, ‘Pavel owes Hikaru money *and* no one besides Pavel owes Hikaru money.’ We have already translated one of the conjuncts, and the other is straightforward. Sentence 38 becomes $owes(P, H) \wedge \neg\exists x, (x \neq P \wedge owes(x, H))$.

Expressions of quantity

We can also use identity to say how many things there are of a particular kind. For example, consider these sentences:

39. There is at least one apple on the table.
40. There are at least two apples on the table.
41. There are at least three apples on the table.

Let the UD be *things on the table*, and let $apple(x)$ mean 'x is an apple.'

Sentence 39 does not require identity. It can be translated adequately as $\exists x, apple(x)$: There is some apple on the table— perhaps many, but at least one.

It might be tempting to also translate sentence 40 without identity. Yet consider the sentence $\exists x, \exists y, (apple(x) \wedge apple(y))$. It means that there is some apple x in the UD and some apple y in the UD. Since nothing precludes x and y from picking out the same member of the UD, this would be true even if there were only one apple. In order to make sure that there are two *different* apples, we need an identity predicate. Sentence 40 needs to say that the two apples that exist are not identical, so it can be translated as $\exists x, \exists y, (apple(x) \wedge apple(y) \wedge x \neq y)$.

Sentence 41 requires talking about three different apples. It can be translated as $\exists x, \exists y, \exists z, (apple(x) \wedge apple(y) \wedge apple(z) \wedge x \neq y \wedge y \neq z \wedge x \neq z)$.

Continuing in this way, we could translate 'There are at least n apples on the table.' There is a summary of how to symbolize sentences like these on p. 200.

Now consider these sentences:

42. There is at most one apple on the table.
43. There are at most two apples on the table.

Sentence 42 can be paraphrased as, 'It is not the case that there are at least *two* apples on the table.' This is just the negation of sentence 40:

$$\neg\exists x, \exists y, (apple(x) \wedge apple(y) \wedge x \neq y)$$

Sentence 42 can also be approached in another way. It means that any apples that there are on the table must be the selfsame apple, so it can be translated as $\forall x, \forall y, \big((apple(x) \wedge apple(y)) \Rightarrow x = y\big)$. The two translations are logically equivalent, so both are correct.

In a similar way, sentence 43 can be translated in two equivalent ways. It can be paraphrased as, 'It is not the case that there are *three* or more distinct apples', so it can be translated as the negation of sentence 41. Using universal quantifiers, it can also be translated as

$$\forall x, \forall y, \forall z, \big((apple(x) \wedge apple(y) \wedge apple(z)) \Rightarrow (x = y \vee x = z \vee y = z)\big).$$

See p. 200 for the general case.

The examples above are sentences about apples, but the logical structure of the sentences translates mathematical inequalities like $a \geq 3$, $a \leq 2$, and so on. We also want to be able to translate statements of equality which say exactly how many things there are. For example:

44. There is exactly one apple on the table.

45. There are exactly two apples on the table.

Sentence 44 can be paraphrased as, 'There is *at least* one apple on the table, and there is *at most* one apple on the table.' This is just the conjunction of sentence 39 and sentence 42: $\exists x, apple(x) \wedge \forall x, \forall y, \big((apple(x) \wedge apple(y)) \Rightarrow x = y\big)$. This is a somewhat complicated way of going about it. It is perhaps more straightforward to paraphrase sentence 44 as, 'There is a thing which is the only apple on the table.' Thought of in this way, the sentence can be translated $\exists x, \big(apple(x) \wedge \neg\exists y, (apple(y) \wedge x \neq y)\big)$.

Similarly, sentence 45 may be paraphrased as, 'There are two different apples on the table, and these are the only apples on the table.' This can be translated as $\exists x, \exists y, \big(apple(x) \wedge apple(y) \wedge x \neq y \wedge \neg\exists z, (apple(z) \wedge x \neq z \wedge y \neq z)\big)$.

Finally, consider this sentence:

46. There are at most two things on the table.

It might be tempting to add a predicate so that $table(x)$ would mean 'x is a thing on the table.' However, this is unnecessary. Since the UD is the set of things on the table, all members of the UD are on the table. If we want to talk about a *thing on the table*, we need only use a quantifier. Sentence 46 can be symbolized like sentence 43 (which said that there were at most two apples), but leaving out the predicate entirely. That is, sentence 46 can be translated as $\forall x, \forall y, \forall z, (x = y \vee x = z \vee y = z)$.

Techniques for symbolizing expressions of quantity ('at most', 'at least', and 'exactly') are summarized on p. 200.

Definite descriptions

Recall that a constant of QL must refer to some member of the UD. This constraint allows us to avoid the problem of non-referring terms. Given a UD that included only actually existing creatures but a constant C that meant 'chimera' (a mythical creature), sentences containing C would become impossible to evaluate.

The most widely influential solution to this problem was introduced by Bertrand Russell in 1905. Russell asked how we should understand this sentence:

47. The present king of France is bald.

The phrase 'the present king of France' is supposed to pick out an individual by means of a definite description. However, there was no king of France in 1905 and there is none now. Since the description is a non-referring term, we cannot just define a constant K to mean 'the present king of France' and translate the sentence as $bald(K)$.

Russell's idea was that sentences that contain definite descriptions have a different logical structure than sentences that contain proper names, even though they share the same grammatical form. What do we mean when we use an unproblematic, referring description, like 'the highest peak in Washington state'? We mean that there is such a peak, because we could not talk about it otherwise. We also mean that it is the only such peak. If there was another peak in Washington state of exactly the same height as Mount Rainier, then Mount Rainier would not be *the* highest peak.

According to this analysis, sentence 47 is saying three things. First, it makes an *existence* claim: There is some present king of France. Second, it makes a *uniqueness* claim: This guy is the only present king of France. Third, it makes a claim of *predication*: This guy is bald.

In order to symbolize definite descriptions in this way, we need the identity predicate. Without it, we could not translate the uniqueness claim which (according to Russell) is implicit in the definite description.

Let the UD be *people actually living*, let $king(x)$ mean 'x is the present king of France', and let $bald(x)$ mean 'x is bald.' Sentence 47 can then be translated as $\exists x, \big(king(x) \wedge \neg\exists y, (king(y) \wedge x \neq y) \wedge bald(x)\big)$. This says that there is some guy who is the present king of France, he is the only present king of France, and he is bald.

Understood in this way, sentence 47 is meaningful but false. It says that this guy exists, but he does not.

The problem of non-referring terms is most vexing when we try to translate negations. So consider this sentence:

48. The present king of France is not bald.

According to Russell, this sentence is ambiguous in English. It could mean either of two things:

48a. It is not the case that the present king of France is bald.
48b. The present king of France is non-bald.

Both possible meanings negate sentence 47, but they put the negation in different places.

Sentence 48a is called a WIDE-SCOPE NEGATION, because it negates the entire sentence. It can be translated as $\neg\exists x, \big(king(x) \wedge \neg\exists y, (king(y) \wedge x \neq y) \wedge bald(x)\big)$. This does not say anything about the present king of France, but rather says that some sentence about the present king of France is false. Since sentence 47 if false, sentence 48a is true.

Sentence 48b says something about the present king of France. It says that he lacks the property of baldness. Like sentence 47, it makes an existence claim and a uniqueness claim; it just denies the claim of predication. This is called NARROW-SCOPE NEGATION. It can be translated as $\exists x, \big(king(x) \wedge \neg\exists y, (king(y) \wedge x \neq y) \wedge \neg bald(x)\big)$. Since there is no present king of France, this sentence is false.

Russell's theory of definite descriptions resolves the problem of non-referring terms and also explains why it seemed so paradoxical. Before we distinguished between the wide-scope and narrow-scope negations, it seemed that sentences like 48 should be both true and false. By showing that such sentences are ambiguous, Russell showed that they are true understood one way but false understood another way.

For a more detailed discussion of Russell's theory of definite descriptions, including objections to it, see Peter Ludlow's entry 'descriptions' in *The Stanford Encyclopedia of Philosophy*: Summer 2005 edition, edited by Edward N. Zalta, `http://plato.stanford.edu/archives/sum2005/entries/descriptions/`

Quiz Yourself

- Is $x = y$ a wff of QL? Is it a sentence of QL?
- On which page of this chapter do you find a technique for translating sentences like "There are exactly two dogs in the house" into QL?
- When a sentence involving a definite description is negated and becomes ambiguous, how many different possible meanings did Bertrand Russell claim it had?

Practice Exercises

In all exercises that ask you to translate sentences from English to QL, it may be helpful to ensure that your results are at least valid wffs of QL. *Lurch* can help you with this if you type the wff in *Lurch* and bubble it as a meaningful expression. When you have done so, if *Lurch* calls the expression a 'string,' then you know it sees it only as a meaningless string of symbols, and you have not successfully written a wff of QL. But if it marks it as, for example, a '$\wedge$ expression' or a '$\forall$ quantification,' you know it is a correctly-formed wff.

It may or may not be the correct English translation of the sentence in the exercise— *Lurch* doesn't know about that! But at least you will know it is a wff. This is especially useful in later exercises when you are writing very large wffs.

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

UD:	all animals
$alli(x)$**:**	x is an alligator.
$mon(x)$**:**	x is a monkey.
$rep(x)$**:**	x is a reptile.
$zoo(x)$**:**	x lives at the zoo.
$loves(x, y)$**:**	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 an 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 sadly Bouncer does not reciprocate this love.

Part B These are syllogistic figures identified by Aristotle and his successors, along with their medieval names. Translate each argument into QL. (Since A, B, and C are predicates in the language Aristotle used, but in QL they are constants, you should replace each with something like *apple*, *bear*, *cabbage*, or perhaps *ay*, *bee*, *cee*, as you prefer.)

Barbara All Bs are Cs. All As are Bs. $\therefore$ All As are Cs.

Baroco All Cs are Bs. Some A is not B. $\therefore$ Some A is not C.

Bocardo Some B is not C. All Bs are As. $\therefore$ Some A is not C.

Celantes No Bs are Cs. All As are Bs. $\therefore$ No Cs are As.

Celarent No Bs are Cs. All As are Bs. $\therefore$ No As are Cs.

Cemestres No Cs are Bs. No As are Bs. $\therefore$ No As are Cs.

Cesare No Cs are Bs. All As are Bs. $\therefore$ No As are Cs.

Dabitis All Bs are Cs. Some A is B. $\therefore$ Some C is A.

Darii All Bs are Cs. Some A is B. $\therefore$ Some A is C.

Datisi All Bs are Cs. Some A is B. $\therefore$ Some A is C.

Disamis Some B is C. All As are Bs. $\therefore$ Some A is C.

Ferison No Bs are Cs. Some A is B. $\therefore$ Some A is not C.

Ferio No Bs are Cs. Some A is B. $\therefore$ Some A is not C.

Festino No Cs are Bs. Some A is B. $\therefore$ Some A is not C.

Baralipton All Bs are Cs. All As are Bs. $\therefore$ Some C is A.

Frisesomorum Some B is C. No As are Bs. $\therefore$ Some C is not A.

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

UD:	all animals
$dog(x)$**:**	x is a dog.
$sam(x)$**:**	x likes samurai movies.
$larger(x, y)$**:**	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. Every dog is larger than some dog.
15. There is an animal that is smaller than every dog.
16. If there is an animal that is larger than any dog, then that animal does not like samurai movies.

Part D 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.

$\star$ **Part E** Using the symbolization key given, translate each English-language sentence into QL.

UD:	candies
$choc(x)$**:**	x has chocolate in it.
$mar(x)$**:**	x has marzipan in it.
$sugar(x)$**:**	x has sugar in it.
$tried(x)$**:**	Boris has tried x.
$better(x, y)$**:**	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 F Using the symbolization key given, translate each English-language sentence into QL.

UD:	people and dishes at a potluck
$run(x)$**:**	x has run out.
$table(x)$**:**	x is on the table.
$food(x)$**:**	x is food.
$per(x)$**:**	x is a person.
$likes(x, y)$**:**	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 anyone who likes the people that he likes.

9. If there is a person on the table already, then all of the food must have run out.

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

UD:	people
$dance(x)$:	x dances ballet.
$fem(x)$:	x is female.
$male(x)$:	x is male.
$child(x, y)$:	x is a child of y.
$sib(x, y)$:	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 H Identify which variables are bound and which are free.

1. $\exists x, likes(x, y) \wedge \forall y, likes(y, x)$
2. $\forall x, angry(x) \wedge bad(x)$
3. $\forall x, (angry(x) \wedge bad(x)) \wedge \forall y, (cat(x) \wedge dog(y))$
4. $\forall x, \exists y, \big(related(x, y) \Rightarrow (junk(z) \wedge king(x))\big) \vee related(y, x)$
5. $\forall p, (mon(q) \Leftrightarrow likes(q, p)) \wedge \exists q, likes(r, q)$

⋆ **Part I**

1. Identify which of the following are substitution instances of $\forall x, related(C, x)$: $related(A, C)$, $related(C, A)$, $related(A, A)$, $related(C, B)$, $related(B, C)$, $related(C, C)$, $related(C, D)$, $related(C, x)$
2. Identify which of the following are substitution instances of $\exists x, \forall y, likes(x, y)$:
$\forall y, likes(B, y)$
$\forall x, likes(B, x)$
$likes(A, B)$
$\exists x, likes(x, A)$

Part J Using the symbolization key given, translate each English-language sentence into QL with identity. The last sentence is ambiguous and can be translated two ways; you should provide both translations. (Hint: Identity is only required for the last four sentences.)

UD:	people
$knows(x)$:	x knows the combination to the safe.
$spy(x)$:	x is a spy.
$veg(x)$:	x is a vegetarian.
$trusts(x, y)$:	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 K** Using the symbolization key given, translate each English-language sentence into QL with identity. The last two sentences are ambiguous and can be translated two ways; you should provide both translations for each.

UD:	cards in a standard deck
$black(x)$:	x is black.
$club(x)$:	x is a club.
$deuce(x)$:	x is a deuce.
$jack(x)$:	x is a jack.
$axe(x)$:	x is a man with an axe.
$eye(x)$:	x is one-eyed.
$wild(x)$:	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 two black jacks.
7. There are 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 L Using the symbolization key given, translate each English-language sentence into QL with identity. The last two sentences are ambiguous and can be translated two ways; you should provide both translations for each.

UD:	animals in the world
$brown(x)$**:**	x is in Farmer Brown's field.
$horse(x)$**:**	x is a horse.
$peg(x)$**:**	x is a Pegasus.
$wings(x)$**:**	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.

Chapter 8

Formal semantics

In this chapter, we describe a *formal semantics* for SL and for QL. The word 'semantics' comes from the greek word for 'mark' and means 'related to meaning.' So a formal semantics will be a mathematical account of meaning in the formal language.

A formal, logical language is built from two kinds of elements: logical symbols and non-logical symbols. Connectives (like '$\wedge$') and quantifiers (like '$\forall$') 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 '$\neg$' symbol will mean 'not' in one argument and 'perhaps' in another. The '$\neg$' 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 QL, the predicates and constants are non-logical symbols.

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.

It is possible to provide different interpretations that make no formal difference. In SL, for example, we might say that D means 'Today is Tuesday'; we might say instead that D means 'Today is the day after Monday.' These are two different interpretations, because they use different English sentences for the meaning of D. Yet, formally, there is no difference between them. All that matters once we have symbolized these sentences is whether they are true or false. In order to characterize what makes a difference in the formal language, we need to know what makes sentences true or false. For this, we need a formal characterization of *truth*.

When we gave definitions for a sentence of SL and for a sentence of QL, we distinguished between the OBJECT LANGUAGE and the METALANGUAGE. The object language is the language that we are *talking about*: either SL or QL. The metalanguage is the language that we use to talk about the object language: English, supplemented with some mathematical jargon. It will be important to keep this distinction in mind.

8.1 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: $\mathscr{A}$ is a tautology in SL if it is T 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 T and F 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.

Once we construct a truth table, the symbols 'T' and 'F' are divorced from their metalinguistic meaning of 'true' and 'false'. We *interpret* 'T' as meaning 'true', but the formal properties of T are defined by the characteristic truth tables for the various connectives. The tables would be the same if we had used the symbols '1' and '0', and computers can be programmed to fill out truth tables without having any sense that 1 means true and 0 means false.

Formally, what we want is a function that assigns a T or F to each of the sentences of SL. Just as a function from algebra or calculus, say $f(x) = x^2 + 1$, might take real number inputs and give (in that case) positive real number outputs, we want a function that will take sentences as inputs and give T's and F's as outputs. We can interpret this function as a definition of truth for SL if it assigns T to all of the true sentences of SL and F to all of the false sentences of SL. Call this function v (for 'valuation'). We want v to a be a function such that for any sentence $\mathscr{A}$, $v(\mathscr{A}) = \text{T}$ if $\mathscr{A}$ is true and $v(\mathscr{A}) = \text{F}$ if $\mathscr{A}$ 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 $\mathscr{A}$ is a wff, then $\neg\mathscr{A}$ 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.

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.

Moreover, the way you would discover whether or not M is true depends on what M means. If M means 'It is Monday,' then you would need to check a calendar. If M means 'Jupiter's moon Io has significant volcanic activity,' then you would need to check an astronomy text— and astronomers know because they sent satellites to observe Io.

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.

Consider another example: If M means 'It is morning now', then whether it is true or not depends on when you are reading this. I know what the sentence means, but— since I do not know when you will be reading this— I do not know whether it is true or false.

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.

$$\text{INTERPRETATION} + \text{STATE OF THE WORLD} \Longrightarrow \text{TRUTH/FALSITY}$$

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

$$a(\mathcal{P}) = \begin{cases} \text{T} & \text{if } \mathcal{P} \text{ is true,} \\ \text{F} & \text{otherwise.} \end{cases}$$

This means that a takes any sentence of SL and assigns it either a T or an F; T if the sentence is true, F if the sentence is false. The details of the function a are determined by the meaning of the sentence letters together with the state of the world. If D means 'It is dark outside', then $a(D) = \text{T}$ at night or during a heavy storm, while $a(D) = \text{F}$ on a clear day.

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 $\mathcal{A}$ is a sentence letter, then $v(\mathcal{A}) = a(\mathcal{A})$.
2. If $\mathcal{A}$ is $\neg\mathcal{B}$ for some sentence $\mathcal{B}$, then

$$v(\mathcal{A}) = \begin{cases} \text{T} & \text{if } v(\mathcal{B}) = \text{F,} \\ \text{F} & \text{otherwise.} \end{cases}$$

3. If $\mathcal{A}$ is $(\mathcal{B} \wedge \mathcal{C})$ for some sentences $\mathcal{B}, \mathcal{C}$, then

$$v(\mathcal{A}) = \begin{cases} \text{T} & \text{if } v(\mathcal{B}) = \text{T and } v(\mathcal{C}) = \text{T,} \\ \text{F} & \text{otherwise.} \end{cases}$$

It might seem as if this definition is circular, because it uses the word 'and' in trying to define 'and.' Notice, however, that this is not a definition of the English word 'and'; it is a definition of truth for sentences of SL containing the logical symbol '$\wedge$.' We define truth for object language sentences containing the symbol '$\wedge$' using the metalanguage word 'and.' There is nothing circular about that.

4. If $\mathcal{A}$ is $(\mathcal{B} \vee \mathcal{C})$ for some sentences $\mathcal{B}, \mathcal{C}$, then

$$v(\mathcal{A}) = \begin{cases} \text{F} & \text{if } v(\mathcal{B}) = \text{F and } v(\mathcal{C}) = \text{F}, \\ \text{T} & \text{otherwise.} \end{cases}$$

5. If $\mathcal{A}$ is $(\mathcal{B} \Rightarrow \mathcal{C})$ for some sentences $\mathcal{B}, \mathcal{C}$, then

$$v(\mathcal{A}) = \begin{cases} \text{F} & \text{if } v(\mathcal{B}) = \text{T and } v(\mathcal{C}) = \text{F}, \\ \text{T} & \text{otherwise.} \end{cases}$$

6. If $\mathcal{A}$ is $(\mathcal{B} \Leftrightarrow \mathcal{C})$ for some sentences $\mathcal{B}, \mathcal{C}$, then

$$v(\mathcal{A}) = \begin{cases} \text{T} & \text{if } v(\mathcal{B}) = v(\mathcal{C}), \\ \text{F} & \text{otherwise.} \end{cases}$$

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 truth value assignment, because the above 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 a.

Other concepts in SL

Working with SL so far, we have done without a precise definition of 'tautology', 'contradiction', and so on. Truth tables provided a way to *check if* 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, '$\mathcal{A}$ entails $\mathcal{B}$', means that there is no truth value assignment for which $\mathcal{A}$ is true and $\mathcal{B}$ is false. Put differently, it means that $\mathcal{B}$ is true for any and all truth value assignments for which $\mathcal{A}$ is true.

We abbreviate this with a symbol called the *double turnstile*: $\mathcal{A} \models \mathcal{B}$ means '$\mathcal{A}$ semantically entails $\mathcal{B}$.'

We can talk about entailment between more than two sentences:

$$\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots\} \models \mathcal{B}$$

means that there is no truth value assignment for which all of the sentences in the set $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots\}$ are true and $\mathcal{B}$ is false.

We can also use the symbol with just one sentence: $\models \mathcal{C}$ means that $\mathcal{C}$ is true for all truth value assignments. This is equivalent to saying that the sentence is entailed by anything.

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

A TAUTOLOGY IN SL is a sentence $\mathcal{A}$ such that $\models \mathcal{A}$.

A CONTRADICTION IN SL is a sentence $\mathcal{A}$ such that $\models \neg\mathcal{A}$.

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

An argument '$\mathcal{P}_1, \mathcal{P}_2, \ldots, \therefore \mathcal{C}$' is VALID IN SL if and only if $\{\mathcal{P}_1, \mathcal{P}_2, \ldots\} \models \mathcal{C}$.

Two sentences $\mathcal{A}$ and $\mathcal{B}$ are LOGICALLY EQUIVALENT IN SL if and only if both $\mathcal{A} \models \mathcal{B}$ and $\mathcal{B} \models \mathcal{A}$.

Logical consistency is somewhat harder to define in terms of semantic entailment. Instead, we will define it in this way:

> The set $\{\mathscr{A}_1, \mathscr{A}_2, \mathscr{A}_3, \ldots\}$ is CONSISTENT IN SL if and only if there is at least one truth value assignment for which all of the sentences are true. The set is INCONSISTENT IN SL if and if only there is no such assignment.

Quiz Yourself

- What is the difference between an interpretation and a valuation?
- We had already learned a semantics for SL before this chapter. What name did we give it?
- What does the symbol $\vDash$ mean?

8.2 Interpretations and models in QL

In SL, an interpretation or symbolization key specifies what each of the sentence letters means. The interpretation of a sentence letter along with the state of the world determines whether the sentence letter is true or false. Since the basic units are sentence letters, an interpretation only matters insofar as it makes sentence letters true or false. Formally, the semantics for SL is strictly in terms of truth value assignments. Two interpretations are the same, formally, if they make for the same truth value assignment.

What is an interpretation in QL? Like a symbolization key for QL, an interpretation requires a UD, a schematic meaning for each of the predicates, and an object that is picked out by each constant. For example:

UD:	comic book characters
$fc(x)$**:**	x fights crime.
B**:**	the Batman
W**:**	Bruce Wayne

Consider the sentence $fc(B)$. The sentence is true on this interpretation, but— just as in SL— the sentence is not true *just because* of the interpretation. Most people in our culture know that Batman fights crime, but this requires a modicum of knowledge about comic books or movies. The sentence $fc(B)$ is true because of the interpretation *plus* some facts about comic books or movies. This is especially obvious when we consider $fc(W)$. Bruce Wayne is the secret identity of the Batman in the comic books— the identity claim $B = W$ is true— so $fc(W)$ is true. Since it is a *secret* identity, however, other characters do not know that $fc(W)$ is true even though they know that $fc(B)$ is true.

We could try to characterize this as a truth value assignment, as we did for SL. The truth value assignment would assign T or F to each atomic wff: $fc(B)$, $fc(W)$, and so on. If we were to do that, however, we might just as well translate the sentences from QL to SL by replacing $fc(B)$ and $fc(W)$ 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 $fc(B)$ and $fc(W)$. Instead, we give meanings to fc, B, and W. This is essential because we want to be able to use quantifiers. There is no adequate way to translate $\forall x, fc(x)$ into SL.

So we want a formal counterpart to an interpretation for predicates and constants, not just for sentences. We cannot use a truth value assignment for this, because a predicate is neither true nor false. In the interpretation given above, fc is true *of* the Batman (i.e., $fc(B)$ is true), but it makes no sense at all to ask whether fc on its own is true. It would be like asking whether the English language fragment '... fights crime' 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 $fc(x)$ to mean 'x fights crime' picks out Batman, Superman, Spiderman, and other heroes as the things that fit fc. Formally, this is a set of members of the UD to which the predicate applies; this set is called the EXTENSION of the predicate.

Many predicates have indefinitely large extensions. It would be impractical to try to write down all of the comic book crime fighters 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 Black Lightning fights crime, for instance), you need to know about comic books. 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 lwm to the key above. We want $lwm(x)$ to mean 'x lives in Wayne Manor', so we write the extension as a set of characters:

extension(lwm) = {Bruce Wayne, Alfred the butler, Dick Grayson}

You do not need to know anything about comic books to be able to determine that, on this interpretation, $lwm(W)$ is true: Bruce Wayne is just specified to be one of the things that is lwm. Similarly, $\exists x, lwm(x)$ is obviously true on this interpretation: There is at least one member of the UD that satisfies the predicate lwm— in fact, there are three of them.

What about the sentence $\forall x, lwm(x)$? The sentence is false, because it is not true that all members of the UD satisfy the predicate lwm. It requires the barest minimum of knowledge about comic books to know that there are other characters besides just these three. Although we specified the extension of lwm 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 B and W? 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 B and W 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 'Batman' and 'Bruce Wayne' to refer to the same comic book character. In this interpretation, we can use the different constants 'B' and 'W' to refer to the same member of the UD.

Sets

We use curly brackets '{' and '}' to denote sets. The members of the set can be listed in any order, separated by commas. The fact that sets can be in any order is important, because it 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 $\emptyset$.

Models

As we have seen, an interpretation in QL is formally significant only 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:	People who played as part of the Three Stooges
$hhh(x)$**:**	x had head hair.
F**:**	Mister Fine

If you do not know anything about the Three Stooges, you will not be able to say which sentences of QL are true on this interpretation. Perhaps you just remember Larry, Curly, and Moe. Is the sentence $hhh(F)$ true or false? It depends on which of the stooges is Mister Fine.

What is the model that corresponds to this interpretation? There were six people who played as part of the Three Stooges over the years, so the UD will have six members: Larry Fine, Moe Howard, Curly Howard, Shemp Howard, Joe Besser, and Curly Joe DeRita. Curly, Joe, and Curly Joe were the only completely bald stooges. The result is this model:

$$\begin{aligned} \text{UD} &= \{\text{Larry, Curly, Moe, Shemp, Joe, Curly Joe}\} \\ \text{extension}(hhh) &= \{\text{Larry, Moe, Shemp}\} \\ \text{referent}(F) &= \text{Larry} \end{aligned}$$

You do not need to know anything about the Three Stooges in order to evaluate whether sentences are true or false in this *model*. The sentence $hhh(F)$ is true, since the referent of F (Larry) is in the extension of hhh. Both $\exists x, hhh(x)$ and $\exists x, \neg hhh(x)$ are true, since there is at least one member of the UD that is in the extension of hhh and at least one member that is not in the extension of hhh. In this way, the model captures all of the formal significance of the interpretation.

Now consider this interpretation:

UD:	positive whole numbers less than 10
$even(x)$**:**	x is even.
$neg(x)$**:**	x is negative.
$less(x, y)$**:**	x is less than y.
$te(x, y, z)$**:**	x times y equals z.

What is the model that goes with this interpretation? The UD is the set $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

The extension of a one-place predicate like *even* or *neg* is just the subset of the UD of which the predicate is true. Roughly speaking, the extension of the predicate *even* is the set of *even*s in the UD. The extension of *even* is the subset $\{2, 4, 6, 8\}$. There are many even numbers besides these four, but these are the only members of the UD that are even. There are no negative numbers in the UD, so *neg* has an empty extension; i.e. extension(neg) $= \emptyset$.

The extension of a two-place predicate like *less* is somewhat vexing. It seems as if the extension of *less* ought to contain 1, since 1 is less than all the other numbers; it ought to contain 2, since 2 is less than all of the other numbers besides 1; and so on. Every member of the UD besides 9 is less than some member of the UD. What would happen if we just wrote extension($less$) $= \{1, 2, 3, 4, 5, 6, 7, 8\}$?

The problem is that sets can be written in any order, so this would be the same as writing extension($less$) $= \{8, 7, 6, 5, 4, 3, 2, 1\}$. This does not tell us which of the members of the set are less than which other members.

We need some way of showing that 1 is less than 8 but that 8 is not less than 1. The solution is to have the extension of *less* consist of pairs of numbers. An ORDERED PAIR is like a set with two members, except that the order *does* matter. We write ordered pairs with parentheses '(' and ')'. The ordered pair (foo, bar) is different than the ordered pair (bar, foo). This notation should seem familiar from algebra and geometry, where ordered pairs are used to represent points in the plane. The extension of *less* 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(*less*) = {(1,2), (1,3), (1,4), (1,5), (1,6), (1,7), (1,8), (1,9), (2,3), (2,4), (2,5), (2,6), (2,7), (2,8), (2,9), (3,4), (3,5), (3,6), (3,7), (3,8), (3,9), (4,5), (4,6), (4,7), (4,8), (4,9), (5,6), (5,7), (5,8), (5,9), (6,7), (6,8), (6,9), (7,8), (7,9), (8,9)}

Three-place predicates will work similarly; 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 *te* in this model will contain ordered triples like (2,4,8), because $2 \times 4 = 8$.

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 through a_n are members of the UD and the predicate is true of a_1 through a_n in that order.

Quiz Yourself

- What does the symbol $\emptyset$ mean?
- What is the difference between a model and a symbolization key?

8.3 Semantics for identity

Identity is a special predicate of QL. We write it a bit differently than other two-place predicates: $x = y$ instead of $id(x, y)$. We also do not need to include it in a symbolization key. The sentence $x = y$ always means 'x is identical to y,' and it cannot be interpreted to mean anything else. In the same way, when you construct a model, you do not get to pick and choose which ordered pairs go into the extension of the identity predicate. It always contains just the ordered pairs of each object in the UD with itself.

The sentence $\forall x, id(x, x)$, which contains an ordinary two-place predicate *id*, is contingent. Whether it is true for an interpretation depends on how you interpret *id*, and whether it is true in a model depends on the extension of *id*. (Again, there is nothing special about the predicate *id*; we are just mentioning it here to contrast it with =.)

In contrast, the sentence $\forall x, x = x$ is a tautology. The extension of identity will always make it true.

Notice that although identity always has the same interpretation, it does not always have the same extension. The extension of identity depends on the UD. If the UD in a model is the set {Doug}, then extension(=) in that model is {(Doug, Doug)}. If the UD is the set {Doug, Omar}, then extension(=) in that model is {(Doug, Doug), (Omar, Omar)}. And so on.

If the referent of two constants is the same, then anything which is true of one is true of the other. For example, if referent(A) = referent(B), then $angry(A) \Leftrightarrow angry(B)$, $bad(A) \Leftrightarrow bad(B)$, $cute(A) \Leftrightarrow cute(B)$, $related(C, A) \Leftrightarrow related(C, B)$, $\forall x, related(x, A) \Leftrightarrow \forall x, related(x, B)$, and so on for any two sentences containing A and B. However, the reverse is not true.

It is possible that anything which is true of A is also true of B, and yet A and B still have different referents. This may seem puzzling, but it is easy to construct a model that shows this. Consider this model:

$$\begin{aligned}
\text{UD} &= \{\text{Rosencrantz, Guildenstern}\}\\
\text{referent}(A) &= \text{Rosencrantz}\\
\text{referent}(B) &= \text{Guildenstern}\\
\text{for all predicates } \mathscr{P}\text{, extension}(\mathscr{P}) &= \emptyset\\
\text{extension}(=) &= \{(\text{Rosencrantz, Rosencrantz}),\\
&\quad\ (\text{Guildenstern, Guildenstern})\}
\end{aligned}$$

This specifies an extension for every predicate of QL: All the infinitely-many predicates are empty. This means that both $angry(A)$ and $angry(B)$ are false, and they are equivalent; both $bad(A)$ and $bad(B)$ are false; and so on for any two sentences that contain A and B. Yet A and B refer to different things. We have written out the extension of identity to make this clear: The ordered pair $\langle \text{referent}(A), \text{referent}(B)\rangle$ is not in it. In this model, $A = B$ is false and $A \neq B$ is true.

8.4 Working with models

We will use the double turnstile symbol for QL much as we did for SL. '$\mathscr{A} \models \mathscr{B}$' means that '$\mathscr{A}$ entails $\mathscr{B}$': When $\mathscr{A}$ and $\mathscr{B}$ are two sentences of QL, $\mathscr{A} \models \mathscr{B}$ means that there is no model in which $\mathscr{A}$ is true and $\mathscr{B}$ is false. $\models \mathscr{A}$ means that $\mathscr{A}$ is true in every model.

This allows us to give definitions for various concepts in QL. Because we are using the same symbol, these definitions will look similar to the definitions in SL. Remember, however, that the definitions in QL are in terms of *models* rather than in terms of truth value assignments.

There is an analogy here between truth value assignments (truth table rows) for SL and models for QL. When interpreting statements of SL, a truth value assignment tells us a possible state of the world, but when interpreting statements of QL, a model does. Thus the symbol $\models$ talks about all truth value assignments for the SL case (all possible states of the world, for SL) but about all possible models for the QL case. Thus we have the following analogous definitions.

A TAUTOLOGY IN QL is a sentence $\mathscr{A}$ that is true in every model; i.e., $\models \mathscr{A}$.

A CONTRADICTION IN QL is a sentence $\mathscr{A}$ that is false in every model; i.e., $\models \neg\mathscr{A}$.

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

An argument '$\mathscr{P}_1, \mathscr{P}_2, \ldots, \ \therefore\ \mathscr{C}$' 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., $\{\mathscr{P}_1, \mathscr{P}_2, \ldots\} \models \mathscr{C}$. It is INVALID IN QL otherwise.

Two sentences $\mathscr{A}$ and $\mathscr{B}$ are LOGICALLY EQUIVALENT IN QL if and only if both $\mathscr{A} \models \mathscr{B}$ and $\mathscr{B} \models \mathscr{A}$.

The set $\{\mathscr{A}_1, \mathscr{A}_2, \mathscr{A}_3, \ldots\}$ 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 if only there is no such model.

Constructing models

Suppose we want to show that $\forall x, alike(x, x) \Rightarrow bad(D)$ is *not* a tautology. This requires showing that the sentence is not true in every model; i.e., that it is false in some model. If we can provide just one 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 $\forall x, \mathit{alike}(x,x) \Rightarrow \mathit{bad}(D)$ to be false, the antecedent ($\forall x, \mathit{alike}(x,x)$) must be true, and the consequent ($\mathit{bad}(D)$) 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 the city of Paris. Since Paris is the only member of the UD, it must be the referent of D.

We want $\forall x, \mathit{alike}(x,x)$ to be true, so we want all members of the UD to be paired with themself in the extension of *alike*; this means that the extension of *alike* must be {(Paris,Paris)}.

We want $\mathit{bad}(D)$ to be false, so the referent of D (i.e., Paris) must not be in the extension of *bad*. We give *bad* an empty extension. The model we have constructed looks like this:

$$\begin{aligned} \text{UD} &= \{\text{Paris}\} \\ \text{extension}(\mathit{alike}) &= \{(\text{Paris,Paris})\} \\ \text{extension}(\mathit{bad}) &= \emptyset \\ \text{referent}(D) &= \text{Paris} \end{aligned}$$

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 $\forall x, \mathit{alike}(x,x) \Rightarrow \mathit{bad}(D)$ is false. Predicates like *hairy* and constants like F make no difference to the truth or falsity of this sentence. It is enough to specify extensions for *alike* and *bad* and a referent for D, as we have done. This provides a *partial model* in which the sentence is false, just as in SL we provided partial truth value assignments.

Perhaps you are wondering: What does the predicate *alike* mean in English? The partial model could correspond to an interpretation like this one:

UD:	Paris
$\mathit{alike}(x,y)$**:**	x is in the same country as y.
$\mathit{bad}(x)$**:**	x was founded in the 20th century.
D**:**	the City of Lights

However, all that the partial model tells us is that *alike* is a predicate which is true of the pair (Paris, Paris). There are indefinitely many predicates in English that have this extension. So $\mathit{alike}(x,y)$ might instead translate as 'x is the same size as y' or 'x and y are both cities.' Similarly, $\mathit{bad}(x)$ is just some predicate that does not apply to Paris; it might instead translate as 'x is on an island' or 'x is a subcompact car.' When we specify the extensions of *alike* and *bad*, we do not specify what English predicates *alike* and *bad* should be used to translate. We are concerned with whether the statement $\forall x, \mathit{alike}(x,x) \Rightarrow \mathit{bad}(D)$ 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 $\forall x, \mathit{alike}(x,x) \Rightarrow \mathit{bad}(D)$ is not a contradiction. We need only specify a model in which $\forall x, \mathit{alike}(x,x) \Rightarrow \mathit{bad}(D)$ is true; i.e., a model in which either $\forall x, \mathit{alike}(x,x)$ is false or $\mathit{bad}(D)$ is true. Here is one such partial model:

$$\begin{aligned} \text{UD} &= \{\text{Paris}\} \\ \text{extension}(\mathit{alike}) &= \{(\text{Paris,Paris})\} \\ \text{extension}(\mathit{bad}) &= \{\text{Paris}\} \\ \text{referent}(D) &= \text{Paris} \end{aligned}$$

We have now shown that $\forall x, \mathit{alike}(x,x) \Rightarrow \mathit{bad}(D)$ is neither a tautology nor a contradiction. By the definition of 'contingent in QL,' this means that $\forall x, \mathit{alike}(x,x) \Rightarrow \mathit{bad}(D)$ 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 $\forall x, small(x)$ and $\exists x, small(x)$ 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 small so that we can specify extensions easily. We will need at least two members. (If we chose a UD with only one member, the two sentences would end up with the same truth value. In order to see why, try constructing some partial models with one-member UDs.) Let the UD be {Duke, Miles}.

We can make $\exists x, small(x)$ true by including something in the extension of *small*, and we can make $\forall x, small(x)$ false by leaving something out of the extension of *small*. It does not matter which one we include and which one we leave out. Making Duke the only *small*, we get a partial model that looks like this:

$$\begin{aligned} \text{UD} &= \{\text{Duke, Miles}\} \\ \text{extension}(small) &= \{\text{Duke}\} \end{aligned}$$

This partial model shows that the two sentences are *not* logically equivalent.

Back on p. 88, we said that this argument would be invalid in QL:

$$\begin{array}{ll} & (sur(C) \wedge ss(C)) \wedge tennis(C) \\ \therefore & tennis(C) \wedge st(C) \end{array}$$

In order to show that it is invalid, we need to show that there is some model in which the premises are true and the conclusion is false. We can construct such a model deliberately. Here is one way to do it:

$$\begin{aligned} \text{UD} &= \{\text{Björk}\} \\ \text{extension}(sur) &= \{\text{Björk}\} \\ \text{extension}(ss) &= \{\text{Björk}\} \\ \text{extension}(tennis) &= \{\text{Björk}\} \\ \text{extension}(st) &= \emptyset \\ \text{referent}(C) &= \text{Björk} \end{aligned}$$

Similarly, we can show that a set of sentences is consistent by constructing a model in which all of the sentences are true.

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. This cannot be avoided just by constructing partial models, because there are infinitely many partial models.

Consider, for example, the sentence $related(A, A) \Leftrightarrow related(A, A)$. 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, $related(A, A) \Leftrightarrow related(A, A)$ is obviously a tautology. We can prove it with a simple argument:

> There are two kinds of models: those in which $\langle \text{referent}(A), \text{referent}(A)\rangle$ is in the extension of R and those in which it is not. In the first kind of model, $related(A, A)$ is true; by the truth table for the biconditional, $related(A, A) \Leftrightarrow related(A, A)$ is also true. In the second kind of model, $related(A, A)$ is false; this makes $related(A, A) \Leftrightarrow related(A, A)$ true. Since the sentence is true in both kinds of model, and since every model is one of the two kinds, $related(A, A) \Leftrightarrow related(A, A)$ is true in every model. Therefore, it is a tautology.

This argument is valid, of course, and its conclusion is true. However, 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 $\forall x, (related(x, x) \Rightarrow related(x, x))$, another obvious tautology. It might be tempting to reason in this way: '$related(x, x) \Rightarrow related(x, x)$ is true in every model, so $\forall x, (related(x, x) \Rightarrow related(x, x))$ must be true.' The problem is that $related(x, x) \Rightarrow related(x, x)$ 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 $related(x, x) \Rightarrow related(x, x)$. In the next section, we introduce the concept of *satisfaction*; after doing so, we will be better able to provide an argument that $\forall x, (related(x, x) \Rightarrow related(x, x))$ 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 contradition, that two sentences are equivalent, that a set of sentences is inconsistent, or that an argument is valid. But there are other things we can show by carefully constructing a model or two. Table 8.1 summarizes which things are in which category.

8.5 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 simplest atomic sentence of QL consists of a one-place predicate applied to a constant, like $pred(J)$. It is true in a model $\mathbb{M}$ if and only if the referent of J is in the extension of $pred$ 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 $\mathcal{pred}(\mathcal{C}_1, \ldots, \mathcal{C}_n)$ where $\mathcal{pred}$ is an n-place predicate and the $\mathcal{C}$'s are constants. It is true in $\mathbb{M}$ if and only if $\langle \text{referent}(\mathcal{C}_1), \ldots, \text{referent}(\mathcal{C}_n)\rangle$ is in extension($\mathcal{pred}$) 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 $pred(J) \Rightarrow mom(D, A)$ is true in $\mathbb{M}$ if either $pred(J)$ is false in $\mathbb{M}$ or $mom(D, A)$ is true in $\mathbb{M}$.

Unfortunately, this approach will fail when we consider sentences containing quantifiers. Consider $\forall x, pred(x)$. When is it true in a model $\mathbb{M}$? The answer cannot depend on whether $pred(x)$ is true or false in $\mathbb{M}$, because the x in $pred(x)$ is a free variable. Thus $pred(x)$ 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

	YES	NO
Is $\mathcal{A}$ a tautology?	show that $\mathcal{A}$ must be true in any model	$\boxed{\text{\textit{construct a model} in which } \mathcal{A} \text{ is false}}$
Is $\mathcal{A}$ a contradiction?	show that $\mathcal{A}$ must be false in any model	$\boxed{\text{\textit{construct a model} in which } \mathcal{A} \text{ is true}}$
Is $\mathcal{A}$ contingent?	$\boxed{\text{\textit{construct two models}, one in which } \mathcal{A} \text{ is true and another in which } \mathcal{A} \text{ is false}}$	either show that $\mathcal{A}$ is a tautology or show that $\mathcal{A}$ is a contradiction
Are $\mathcal{A}$ and $\mathcal{B}$ equivalent?	show that $\mathcal{A}$ and $\mathcal{B}$ must have the same truth value in any model	$\boxed{\text{\textit{construct a model} in which } \mathcal{A} \text{ and } \mathcal{B} \text{ have different truth values}}$
Is the set $\mathbb{A}$ consistent?	$\boxed{\text{\textit{construct a model} in which all the sentences in } \mathbb{A} \text{ are true}}$	show that the sentences could not all be true in any model
Is the argument '$\mathcal{P}$, $\therefore$ $\mathcal{C}$' valid?	show that any model in which $\mathcal{P}$ is true must be a model in which $\mathcal{C}$ is true	$\boxed{\text{\textit{construct a model} in which } \mathcal{P} \text{ is true and } \mathcal{C} \text{ is false}}$

Table 8.1: 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 (the boxed entries).

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.

Satisfaction

The formula $pred(x)$ says, roughly, that x is one of the *pred* things. 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 $\forall x, pred(x)$, but it only needs to stand in for one member in $\exists x, pred(x)$. Since we want the definition of satisfaction to cover $pred(x)$ without any quantifier whatsoever, we will start by saying how to interpret a free variable like the x in $pred(x)$.

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 $pred(x)$ 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 the extension of *pred* in $\mathbb{M}$.

When is $\forall x, pred(x)$ satisfied? It is not enough if $pred(x)$ is satisfied in $\mathbb{M}$ by a, because that just means that $a(x)$ is in extension($pred$). In addition, $\forall x, pred(x)$ requires that every other member of the UD be in extension($pred$) as well.

So we need another bit of technical notation: For any member Ω of the UD and any variable χ, let $a[\chi = \Omega]$

be the variable assignment that assigns Ω to $\mathcal{X}$ but agrees with a in all other respects. We have used Ω, the Greek letter Omega, 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 $a[x = \text{Grover Cleveland}]$ assigns Grover Cleveland to the variable x, regardless of what a assigns to x; for any other variable, $a[x = \text{Grover Cleveland}]$ agrees with a.

We can now say concisely that $\forall x, pred(x)$ is satisfied in a model $\mathbb{M}$ by a variable assignment a if and only if, for every object Ω in the UD of $\mathbb{M}$, $pred(x)$ is satisfied in $\mathbb{M}$ by $a[x = \Omega]$.

You may worry that this is circular, because it gives the satisfaction conditions for the sentence $\forall x, pred(x)$ 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 $\mathcal{A}$ and variable assignment a, $s(\mathcal{A}, a) = \text{T}$ if $\mathcal{A}$ is satisfied in $\mathbb{M}$ by a; otherwise $s(\mathcal{A}, a) = \text{F}$.

1. If $\mathcal{A}$ is an atomic wff of the form $\mathcal{pred}(t_1, \ldots, t_n)$ with each t_i a term, and each Ω_i is the object picked out by t_i, then
$$s(\mathcal{A}, a) = \begin{cases} \text{T} & \text{if } \langle \Omega_1, \ldots, \Omega_n \rangle \text{ is in extension}(\mathcal{pred}) \text{ in } \mathbb{M}, \\ \text{F} & \text{otherwise.} \end{cases}$$
For each term t_i: If t_i is a constant, then $\Omega_i = \text{referent}(t_i)$. If t_i is a variable, then $\Omega_i = a(t_i)$.

2. If $\mathcal{A}$ is $\neg\mathcal{B}$ for some wff $\mathcal{B}$, then
$$s(\mathcal{A}, a) = \begin{cases} \text{T} & \text{if } s(\mathcal{B}, a) = \text{F}, \\ \text{F} & \text{otherwise.} \end{cases}$$

3. If $\mathcal{A}$ is $(\mathcal{B} \wedge \mathcal{C})$ for some wffs $\mathcal{B}, \mathcal{C}$, then
$$s(\mathcal{A}, a) = \begin{cases} \text{T} & \text{if } s(\mathcal{B}, a) = \text{T and } s(\mathcal{C}, a) = \text{T}, \\ \text{F} & \text{otherwise.} \end{cases}$$

4. If $\mathcal{A}$ is $(\mathcal{B} \vee \mathcal{C})$ for some wffs $\mathcal{B}, \mathcal{C}$, then
$$s(\mathcal{A}, a) = \begin{cases} \text{F} & \text{if } s(\mathcal{B}, a) = \text{F and } s(\mathcal{C}, a) = \text{F}, \\ \text{T} & \text{otherwise.} \end{cases}$$

5. If $\mathcal{A}$ is $(\mathcal{B} \Rightarrow \mathcal{C})$ for some wffs $\mathcal{B}, \mathcal{C}$, then
$$s(\mathcal{A}, a) = \begin{cases} \text{F} & \text{if } s(\mathcal{B}, a) = \text{T and } s(\mathcal{C}, a) = \text{F}, \\ \text{T} & \text{otherwise.} \end{cases}$$

6. If $\mathcal{A}$ is $(\mathcal{B} \Leftrightarrow \mathcal{C})$ for some sentences $\mathcal{B}, \mathcal{C}$, then
$$s(\mathcal{A}, a) = \begin{cases} \text{T} & \text{if } s(\mathcal{B}, a) = s(\mathcal{C}, a), \\ \text{F} & \text{otherwise.} \end{cases}$$

7. If $\mathcal{A}$ is $\forall\mathcal{X}, \mathcal{B}$ for some wff $\mathcal{B}$ and some variable $\mathcal{X}$, then
$$s(\mathcal{A}, a) = \begin{cases} \text{T} & \text{if } s(\mathcal{B}, a[\mathcal{X} = \Omega]) = \text{T for every member } \Omega \text{ of the UD}, \\ \text{F} & \text{otherwise.} \end{cases}$$

8. If $\mathscr{A}$ is $\exists\chi, \mathscr{B}$ for some wff $\mathscr{B}$ and some variable χ, then

$$s(\mathscr{A}, a) = \begin{cases} \text{T} & \text{if } s(\mathscr{B}, a[\chi = \Omega]) = \text{T for at least one member } \Omega \text{ of the UD,} \\ \text{F} & \text{otherwise.} \end{cases}$$

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.

Truth

Consider a simple sentence like $\forall x, \mathit{pred}(x)$. By part 7 in the definition of satisfaction, this sentence is satisfied if $a[x = \Omega]$ satisfies $\mathit{pred}(x)$ in $\mathbb{M}$ for every Ω in the UD. By part 1 of the definition, this will be the case if every Ω is in the extension of *pred*. Whether $\forall x, \mathit{pred}(x)$ 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: $\forall x, \mathit{pred}(x)$ is true if everything in the UD is in the extension of *pred*.

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 $\mathscr{A}$ is TRUE IN $\mathbb{M}$ if and only if some variable assignment satisfies $\mathscr{A}$ in $\mathbb{M}$; $\mathscr{A}$ 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.

Reasoning about all models (reprise)

At the end of section 8.4, we were stymied when we tried to show that $\forall x, (\mathit{related}(x, x) \Rightarrow \mathit{related}(x, x))$ 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 Ω. It must be the case either that $\langle\Omega, \Omega\rangle$ is in the extension of *related* or that it is not. If $\langle\Omega, \Omega\rangle$ is in the extension of *related*, then $\mathit{related}(x, x)$ is satisfied by a variable assignment that assigns Ω to x (by part 1 of the definition of satisfaction); since the consequent of $\mathit{related}(x, x) \Rightarrow \mathit{related}(x, x)$ is satisfied, the conditional is satisfied (by part 5). If $\langle\Omega, \Omega\rangle$ is not in the extension of *related*, then $\mathit{related}(x, x)$ is not satisfied by a variable assignment that assigns Ω to x (by part 1); since the antecedent of $\mathit{related}(x, x) \Rightarrow \mathit{related}(x, x)$ is not satisfied, the conditional is satisfied (by part 5).
>
> In either case, $\mathit{related}(x, x) \Rightarrow \mathit{related}(x, x)$ is satisfied. This is true for any member of the UD, so $\forall x, (\mathit{related}(x, x) \Rightarrow \mathit{related}(x, x))$ is satisfied by any truth value assignment (by part 7). So $\forall x, (\mathit{related}(x, x) \Rightarrow \mathit{related}(x, x))$ 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 *related*, so $\forall x, (\mathit{related}(x, x) \Rightarrow \mathit{related}(x, x))$ 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. 114, for example, we distinguished two kinds of models based on whether or not a specific ordered pair was in extension(*related*). In the argument above, we distinguished cases in which an ordered pair was in extension(*related*) 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 Ω 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 Ω must hold of every member of the UD— if we could show it for Ω, 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 $\forall x, (happy(x) \wedge jolly(x)), \quad \therefore \forall x, happy(x)$ 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 $\forall x, (happy(x) \wedge jolly(x))$ is true. The conjunction $happy(x) \wedge jolly(x)$ is satisfied regardless of what is assigned to x, so $happy(x)$ must be also (by part 3 of the definition of satisfaction). As such, $\forall x, happy(x)$ 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 $\forall x, (happy(x) \wedge jolly(x))$ being true, $\forall x, happy(x)$ must be true in any model in which $\forall x, (happy(x) \wedge jolly(x))$ is true. So $\forall x, (happy(x) \wedge jolly(x)) \models \forall x, happy(x)$.

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?

We might try to formalize our reasoning about models, codifying the divide-and-conquer strategies that we used above. This approach, originally called *semantic tableaux*, was developed in the 1950s by Evert Beth and Jaakko Hintikka. Their tableaux are now more commonly called *truth trees*.

A more traditional approach is to consider deductive arguments as proofs. In the next chapter, we develop a proof system for QL based on the one we already know for SL. As in SL, that proof system will consist of rules that formally distinguish between legitimate and illegitimate arguments— without considering models or the meanings of the symbols.

Quiz Yourself

- How many models do you need to show that a given sentence of QL is contingent?
- How many models do you need to show that a given sentence of QL is not a tautology?
- Why is it not reasonable to use models to show that a given sentence of QL is a tautology?
- Rather than try to deal with all models at once, what strategy will we employ?

Practice Exercises

$\star$ **Part A** Determine whether each sentence is true or false in the model given.

$$\begin{aligned}
\text{UD} &= \{\text{Corwin, Benedict}\}\\
\text{extension}(ant) &= \{\text{Corwin, Benedict}\}\\
\text{extension}(bug) &= \{\text{Benedict}\}\\
\text{extension}(nut) &= \emptyset\\
\text{referent}(C) &= \text{Corwin}
\end{aligned}$$

1. $bug(C)$
2. $ant(C) \Leftrightarrow \neg nut(C)$
3. $nut(C) \Rightarrow (ant(C) \vee bug(C))$
4. $\forall x, ant(x)$
5. $\forall x, \neg bug(x)$
6. $\exists x, (ant(x) \wedge bug(x))$
7. $\exists x, (ant(x) \Rightarrow nut(x))$
8. $\forall x, (nut(x) \vee \neg nut(x))$
9. $\exists x, bug(x) \Rightarrow \forall x, ant(x)$

⋆ **Part B** Determine whether each sentence is true or false in the model given.

$$\begin{aligned}
\text{UD} &= \{\text{Waylan, Willy, Johnny}\}\\
\text{extension}(sing) &= \{\text{Waylan, Willy, Johnny}\}\\
\text{extension}(ws) &= \{\text{Waylan, Willy}\}\\
\text{extension}(ring) &= \{(\text{Waylan, Willy}),(\text{Willy, Johnny}),(\text{Johnny, Waylan})\}\\
\text{referent}(M) &= \text{Johnny}
\end{aligned}$$

1. $\exists x, (ring(x, M) \wedge ring(M, x))$
2. $\forall x, (ring(x, M) \vee ring(M, x))$
3. $\forall x, (sing(x) \Leftrightarrow ws(x))$
4. $\forall x, (ring(x, M) \Rightarrow ws(x))$
5. $\forall x, \big(ws(x) \Rightarrow (sing(x) \wedge ws(x))\big)$
6. $\exists x, ring(x, x)$
7. $\exists x, \exists y, ring(x, y)$
8. $\forall x, \forall y, ring(x, y)$
9. $\forall x, \forall y, (ring(x, y) \vee ring(y, x))$
10. $\forall x, \forall y, \forall z, \big((ring(x, y) \wedge ring(y, z)) \Rightarrow ring(x, z)\big)$

Part C Determine whether each sentence is true or false in the model given.

$$\begin{aligned}
\text{UD} &= \{\text{Lemmy, Courtney, Eddy}\}\\
\text{extension}(cool) &= \{\text{Lemmy, Courtney, Eddy}\}\\
\text{extension}(fem) &= \{\text{Courtney}\}\\
\text{extension}(male) &= \{\text{Lemmy, Eddy}\}\\
\text{referent}(C) &= \text{Courtney}\\
\text{referent}(E) &= \text{Eddy}
\end{aligned}$$

1. $fem(C)$
2. $fem(E)$
3. $male(C) \vee male(E)$
4. $cool(C) \vee \neg cool(C)$
5. $male(C) \Rightarrow cool(C)$
6. $\exists x, fem(x)$
7. $\forall x, fem(x)$
8. $\exists x, \neg male(x)$
9. $\exists x, (fem(x) \wedge cool(x))$
10. $\exists x, (male(x) \wedge cool(x))$
11. $\forall x, (fem(x) \vee male(x))$
12. $\exists x, fem(x) \wedge \exists x, male(x)$
13. $\forall x, (fem(x) \Leftrightarrow \neg male(x))$
14. $\exists x, cool(x) \wedge \exists x, \neg cool(x)$

15. $\forall x, \exists y, (cool(x) \wedge fem(y))$

⋆ **Part D** Write out the model that corresponds to the interpretation given.

UD:	natural numbers from 10 to 13 (including both 10 and 13)
$odd(x)$**:**	x is odd.
$ls(x)$**:**	x is less than 7.
$td(x)$**:**	x is a two-digit number.
$tu(x)$**:**	x is thought to be unlucky.
$nna(x, y)$**:**	x is the next number after y.

Part E Show that each of the following is contingent.

⋆ 1. $dog(A) \wedge dog(B)$
⋆ 2. $\exists x, tall(x, H)$
⋆ 3. $pretty(M) \wedge \neg\forall x, pretty(x)$
4. $\forall z, joking(z) \Leftrightarrow \exists y, joking(y)$
5. $\forall x, (wow(x, M, N) \vee \exists y, less(x, y))$
6. $\exists x, (guitar(x) \Rightarrow \forall y, male(y))$

⋆ **Part F** Show that the following pairs of sentences are not logically equivalent.

1. $joking(A), \quad kidding(A)$
2. $\exists x, joking(x), \quad joking(M)$
3. $\forall x, related(x, x), \quad \exists x, related(x, x)$
4. $\exists x, pretty(x) \Rightarrow quiet(C), \quad \exists x, (pretty(x) \Rightarrow quiet(C))$
5. $\forall x, (pretty(x) \Rightarrow \neg quiet(x)), \quad \exists x, (pretty(x) \wedge \neg quiet(x))$
6. $\exists x, (pretty(x) \wedge quiet(x)), \quad \exists x, (pretty(x) \Rightarrow quiet(x))$
7. $\forall x, (pretty(x) \Rightarrow quiet(x)), \quad \forall x, (pretty(x) \wedge quiet(x))$
8. $\forall x, \exists y, related(x, y), \quad \exists x, \forall y, related(x, y)$
9. $\forall x, \exists y, related(x, y), \quad \forall x, \exists y, related(y, x)$

Part G Show that the following sets of sentences are consistent.

1. $\{male(A), \quad \neg nice(A), \quad pretty(A), \quad \neg quiet(A)\}$
2. $\{less(E, E), \quad less(E, F), \quad \neg less(F, E), \quad \neg less(F, F)\}$
3. $\{\neg(male(A) \wedge \exists x, angry(x)), \quad male(A) \vee fast(A), \quad \forall x, (fast(x) \Rightarrow angry(x))\}$
4. $\{male(A) \vee male(B), \quad male(A) \Rightarrow \forall x, \neg male(x)\}$
5. $\{\forall y, guitar(y), \quad \forall x, (guitar(x) \Rightarrow hairy(x)), \quad \exists y, \neg itchy(y)\}$
6. $\{\exists x, (bad(x) \vee angry(x)), \quad \forall x, \neg cute(x), \quad \forall x, \big((angry(x) \wedge bad(x)) \Rightarrow cute(x)\big)\}$
7. $\{\exists x, ax(x), \quad \exists x, ay(x), \quad \forall x, (ax(x) \Leftrightarrow \neg ay(x))\}$
8. $\{\forall x, (pretty(x) \vee quiet(x)), \quad \exists x, \neg(quiet(x) \wedge pretty(x))\}$
9. $\{\exists z, (nut(z) \wedge older(z, z)), \quad \forall x, \forall y, (older(x, y) \Rightarrow older(y, x))\}$
10. $\{\neg\exists x, \forall y, related(x, y), \quad \forall x, \exists y, related(x, y)\}$

Part H Construct models to show that the following arguments are invalid.

1. $\forall x, (angry(x) \Rightarrow bad(x)), \quad \therefore \exists x, bad(x)$
2. $\forall x, (red(x) \Rightarrow dog(x)), \quad \forall x, (red(x) \Rightarrow fast(x)), \quad \therefore \exists x, (dog(x) \wedge fast(x))$
3. $\exists x, (pretty(x) \Rightarrow quiet(x)), \quad \therefore \exists x, pretty(x)$

4. $nut(A) \wedge nut(B) \wedge nut(C), \quad \therefore \forall x, nut(x)$
5. $related(D, E), \quad \exists x, related(x, D), \quad \therefore related(E, D)$
6. $\exists x, (easy(x) \wedge fast(x)), \quad \exists x, fast(x) \Rightarrow \exists x, guitar(x), \quad \therefore \exists x, (easy(x) \wedge guitar(x))$
7. $\forall x, older(x, C), \quad \forall x, older(C, x), \quad \therefore \forall x, older(x, x)$
8. $\exists x, (jack(x) \wedge king(x)), \quad \exists x, \neg king(x), \quad \exists x, \neg jack(x), \quad \therefore \exists x, (\neg jack(x) \wedge \neg king(x))$
9. $less(A, B) \Rightarrow \forall x, less(x, B), \quad \exists x, less(x, B), \quad \therefore less(B, B)$

Part I

⋆ 1. Show that $\{\neg related(A, A), \quad \forall x, (x = A \vee related(x, A))\}$ is consistent.
⋆ 2. Show that $\{\forall x, \forall y, \forall z, (x = y \vee y = z \vee x = z), \quad \exists x, \exists y, x \neq y\}$ is consistent.
⋆ 3. Show that $\{\forall x, \forall y, x = y, \quad \exists x, x \neq A\}$ is inconsistent.
4. Show that $\exists x, (x = H \wedge x = I)$ is contingent.
5. Show that $\{\exists x, \exists y, (zero(x) \wedge zero(y) \wedge x = y), \quad \neg zero(D), \quad D = S\}$ is consistent.
6. Show that '$\forall x, (dead(x) \Rightarrow \exists y, tall(y, x)), \quad \therefore \exists y, \exists z, y \neq z$' is invalid.

Part J

1. Many logic books define consistency and inconsistency in this way: 'A set $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots\}$ is inconsistent if and only if $\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots\} \models (\mathcal{B} \wedge \neg\mathcal{B})$ for some sentence $\mathcal{B}$. A set is consistent if it is not inconsistent.'

   Does this definition lead to any different sets being consistent than the definition on p. 107? Explain your answer.
⋆ 2. Our definition of truth says that a sentence $\mathcal{A}$ is TRUE IN $\mathbb{M}$ if and only if some variable assignment satisfies $\mathcal{A}$ in $\mathbb{M}$. Would it make any difference if we said instead that $\mathcal{A}$ is TRUE IN $\mathbb{M}$ if and only if *every* variable assignment satisfies $\mathcal{A}$ in $\mathbb{M}$? Explain your answer.

Chapter 9

Proofs in QL

In some sense, learning proofs in SL wasn't strictly necessary, because we could check the truth values of sentences, the validity of arguments, and do many other things using truth tables alone. Of course, proofs gave us several advantages over truth tables, as Chapter 5 discussed, but they weren't strictly *necessary,* to accomplish the tasks just listed.

In QL, however, proofs are necessary. Even in the case where one could check the truth value of a sentence by considering only finitely many models, the number of models one would have to check is often prohibitively enormous, koch more so than the number of rows in most truth tables. Secondly, in some cases, the number of possible models is even infinite. Thus we cannot accomplish some goals (showing a QL sentence to be a tautology, showing a set of QL sentences consistent, etc.) without being able to do proofs in QL.

9.1 Rules for quantifiers

For proofs in QL, we use all of the basic rules of SL plus four new basic rules: both introduction and elimination rules for each of the quantifiers.

Since all of the derived rules of SL are derived from the basic rules, they will also hold in QL. We will add another derived rule, a replacement rule called quantifier negation.

Universal elimination

If you have $\forall x, apple(x)$, it is legitimate to infer that anything is an *apple*. You can infer $apple(A)$, $apple(B)$, $apple(Z)$, $apple(R)$— in short, you can infer $apple(\boldsymbol{t})$ for any term $\boldsymbol{t}$. This is the general form of the universal elimination rule (∀E):

$$
\begin{array}{lll}
m. & \forall \chi, \mathcal{A} & \\
 & \mathcal{A}[\chi = \boldsymbol{t}] & \forall\text{E } m.
\end{array}
$$

$\mathcal{A}[\chi = \mathcal{c}]$ is a substitution instance of $\forall \chi, \mathcal{A}$. The symbols for a substitution instance are not symbols of QL, so you cannot write them in a proof. Instead, you write the substituted sentence with the term $\boldsymbol{t}$ replacing all occurrences of the variable χ in $\mathcal{A}$. For example:

1.	$\forall x, (man(x) \Rightarrow related(x, D))$	
2.	$man(A) \Rightarrow related(A, D)$	∀E 1.
3.	$man(D) \Rightarrow related(D, D)$	∀E 1.

Existential introduction

When is it legitimate to infer $\exists x, apple(x)$? If you know that something is an *apple*— for instance, if you have $apple(A)$ available in the proof.

This is the existential introduction rule (∃I):

$m.$	$\mathcal{A}[\chi = t]$	
	$\exists \chi, \mathcal{A}$	∃I $m.$

In other words, if you have already established a substitution instance of $\mathcal{A}$ in your proof, you can conclude $\exists x, \mathcal{A}$. For example:

1.	$man(A) \Rightarrow related(A, D)$	given
2.	$\exists x, (man(A) \Rightarrow related(A, x))$	∃I 1.
3.	$\exists x, (man(x) \Rightarrow related(x, D))$	∃I 1.
4.	$\exists x, (man(x) \Rightarrow related(A, D))$	∃I 1.
5.	$\exists y, \exists x, (man(x) \Rightarrow related(y, D))$	∃I 4.
6.	$\exists z, \exists y, \exists x, (man(x) \Rightarrow related(y, z))$	∃I 5.

Note in particular the difference between lines 3 and 4 above. Line 3 replaced all A's with x's, but line 4 only replaced one. But in each case, line 1 was a substitution instance of the scope of the quantifier, so the step was legal according to the ∃I rule.

Universal introduction

A universal claim like $\forall x, pred(x)$ would be proven if every substitution instance of it had been proven, if every sentence $pred(A)$, $pred(B)$, ... were available in a proof. Alas, there is no hope of proving *every* substitution instance. That would not only require proving $pred(A)$, $pred(B)$, ..., $pred(J)$, ..., $pred(Z)$, but of course there is no restriction that QL can only reason about situations in which there are 26 or fewer things, so knowing something is true about all constants doesn't mean it's true about everything. Furthermore, most logical systems permit infinitely many constants (and in fact *Lurch* does as well); the restriction to A through Z in QL was just a way to make the language simpler to describe and write.

Consider a simple argument: $\forall x, man(x), \quad \therefore \forall y, man(y)$

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:

1.	$\forall x, man(x)$	want $\forall y, man(y)$
2.	$man(A)$	∀E 1.

We have derived $man(A)$. Nothing stops us from using the same justification to derive $man(B)$, ..., $man(J)$, ..., $man(Z)$, or the same statement about any of the other constants in QL, or about new constants we could add. We have effectively shown the way to prove $man(\boldsymbol{t})$ for any term $\boldsymbol{t}$. This is because A was just some arbitrary constant. We had not made any special assumptions about it. If $man(A)$ were a premise of the argument, then this would not show anything about *all y*. For example:

1.	$\forall x, related(x, A)$	
2.	$related(A, A)$	$\forall$E 1.

We could *not* now conclude that $related(\boldsymbol{t}, \boldsymbol{t})$ was true for every $\boldsymbol{t}$. For example, maybe $related(B, B)$ and maybe not! We had assumptions about A, and so the universal introduction rule requires you to work with an *arbitrary variable,* one about which you have no assumptions. Its schematic formula looks like this.

m.	Let $\boldsymbol{a}$ be arbitrary.	
n.	$\mathcal{A}$	
	$\forall\mathcal{X}, \mathcal{A}[\boldsymbol{a} = \mathcal{X}]$	$\forall$I m., n.

Line m is called *declaring* the variable a. No reason need be given for a line that declares a variable, but declaring a variable starts an (indented) subproof. Once a variable is declared, it stays declared until the end of that subproof, and may not be redeclared by any inner subproof. Subproofs that start with a variable declaration end when the $\forall$I rule is invoked, thus finishing our use of the variable.

Thus we can complete our earlier proof as follows.

1.	$\forall x, man(x)$	
2.	Let a be arbitrary.	
3.	$man(a)$	$\forall$E 1.
4.	$\forall y, man(y)$	$\forall$I 2., 3.

As a second example, we can prove $\forall z, (dog(z) \Rightarrow dog(z))$ without any premises. In this example, we use the same variable in the universal quantifier we create on line 5 as we declared to be arbitrary on line 1. This is not strictly necessary, as we saw in the examples above, but it is the most common way that the universal introduction rule is used in mathematical proofs.

1.	Let z be arbitrary.	
2.	$dog(z)$	want $dog(z)$
3.	$dog(z)$	R 2.
4.	$dog(z) \Rightarrow dog(z)$	$\Rightarrow$I 2., 3.
5.	$\forall z, (dog(z) \Rightarrow dog(z))$	$\forall$I 1., 4.

Notice that in neither of these proofs do we end with a free variable in the final line. Recall that statements with free variables in them are not sentences of QL; they only have meanings in subproofs that have declared the variable to be arbitrary. But outside that subproof, where the variable is undeclared, a statement containing the variable free would still have no meaning. Thus your proofs must end with sentences of QL, that is, statements without free variables.

Existential elimination

A sentence with an existential quantifier tells us that there is *some* member of the UD that satisfies a formula. For example, $\exists x, sly(x)$ tells us (roughly) that there is at least one *sly* thing. It does not tell us *which* member of the UD satisfies *sly*, however. We cannot immediately conclude $sly(A)$, $sly(F)$, or any other substitution instance of the sentence that claims that a particular constant is *sly*.

But we know *something* is *sly*, so let's create a name for this thing. We introduce a way of declaring constants, slightly different from how we declared arbitrary variables, as in the ∀I rule. Now we're declaring constants that we're making a specific claim about, based on an existential statement we already know to be true.

This is the schematic form of the existential elimination rule (∃E):

$m.$	$\exists \mathcal{X}, \mathcal{A}$	
$n.$	Let $\mathcal{C}$ be such that:	
	$\mathcal{A}[\mathcal{X} = \mathcal{C}]$	∃E $m.$, $n.$

This new type of declaration says that we are using the constant $\mathcal{C}$ to refer to the thing we know exists, and satisfies the statement $\mathcal{A}$. It cannot be used if the constant $\mathcal{C}$ it's introducing has already appeared somewhere in the proof. You must be borrowing an unused constant, to give a name to the object described by line m, so the $\mathcal{C}$ must not have been mentioned in the proof so far. So a constant declaration is just like a variable declaration in that you cannot redeclare something already in use; you can only declare something new.

The two lines introduced by this rule are a constant declaration and a statement, respectively. The constant declaration cannot be cited later as a premise; only the statement can. The constant declaration line does not require a reason, just as variable declarations do not require them. The constant declaration must occur immediately before the statement, because they form one sentence together.

With this rule, we can give a formal proof that uses an existential as a premise, such as $\exists x, sly(x)$.

1.	$\exists x, sly(x)$	given
2.	$\forall x, (sly(x) \Rightarrow tux(x))$	given; want $\exists x, tux(x)$
3.	Let C be such that:	
4.	$sly(C)$	∃E 1., 3.
5.	$sly(C) \Rightarrow tux(C)$	∀E 2.
6.	$tux(C)$	⇒E 4., 5.
7.	$\exists x, tux(x)$	∃I 6.

Notice that line 5 cites line 3 as a premise, and uses it as if it contained only the statement $sly(C)$, ignoring the constant declaration in line 3.

No proof may contain in its final line a constant that was only declared temporarily within the proof itself. Thus, for example, we could *not* stop the above proof at line 5 and conclude $\exists x, sly(x) \vdash sly(C)$. It should be clear from reading such a statement that it is false; just because *sly* is true of something doesn't mean it must be true of C. In the above proof, C was simply a temporary name borrowed for use in that proof, starting at line 3. Thus it must not be used outside the proof, and so the proof's conclusion cannot contain C.

Quantifier negation

When translating from English to QL, we noted that $\neg\exists x, \neg\mathcal{A}$ is logically equivalent to $\forall x, \mathcal{A}$. In QL, they are provably equivalent. We can prove one half of the equivalence with a rather gruesome proof:

1.	$\forall x, apple(x)$	given; want $\neg\exists x, \neg apple(x)$
2.	$\quad\exists x, \neg apple(x)$	for reductio
3.	$\quad\quad\forall x, apple(x)$	for reductio
4.	$\quad\quad$Let A be such that:	
5.	$\quad\quad\neg apple(A)$	$\exists$E 2., 4.
6.	$\quad\quad apple(A)$	$\forall$E 1.
7.	$\quad\neg\forall x, apple(x)$	$\neg$I 3., 6., 5.
8.	$\quad\forall x, apple(x)$	R 1.
9.	$\neg\exists x, \neg apple(x)$	$\neg$I 2., 8., 7.

In order to show that the two sentences are genuinely equivalent, we need a second proof that assumes $\neg\exists x, \neg\mathcal{A}$ and derives $\forall x, \mathcal{A}$. 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), and are logical equivalences:

$$\begin{array}{l} \neg\forall\mathcal{x}, \mathcal{A} \Longleftrightarrow \exists\mathcal{x}, \neg\mathcal{A} \\ \neg\exists\mathcal{x}, \mathcal{A} \Longleftrightarrow \forall\mathcal{x}, \neg\mathcal{A} \quad \text{QN} \end{array}$$

9.2 Rules for identity

The identity predicate is not part of QL, but we add it when we need to symbolize certain sentences. For proofs involving identity, we add two rules of proof.

Suppose you know that many things that are true of a are also true of b. For example: $apple(A) \wedge apple(B)$, $banana(A) \wedge banana(B)$, $\neg cherry(A) \wedge \neg cherry(B)$, $date(A) \wedge date(B)$, $\neg enchilada(A) \wedge \neg enchilada(B)$, and so on. This would not be enough to justify the conclusion $A = B$. (See p. 110.) In general, there are no sentences that do not already contain the identity predicate that could justify the conclusion $A = B$. This means that the identity introduction rule will not justify $A = B$ or any other identity claim containing two *different* constants.

However, it is always true that $A = A$. In general, no premises are required in order to conclude that something is identical to itself. So this will be the identity introduction rule, abbreviated =I:

$$\boldsymbol{t} = \boldsymbol{t} \qquad =\text{I}$$

Notice that the =I rule does not require referring to any prior lines of the proof. For any term $\boldsymbol{t}$, you can write $\boldsymbol{t} = \boldsymbol{t}$ on any point with only the =I rule as justification.

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 $related(A, A)$, then you are justified in concluding $related(A, B)$, $related(B, A)$, $related(B, B)$. The identity elimination rule (=E) justifies replacing terms with other terms that are identical to it:

$m.$	$\mathcal{a} = \mathcal{b}$	
$n.$	$\mathcal{A}$	
	$\mathcal{A}[\mathcal{b} \sim \mathcal{a}]$	=E $m.$, $n.$
	$\mathcal{A}[\mathcal{a} \sim \mathcal{b}]$	=E $m.$, $n.$

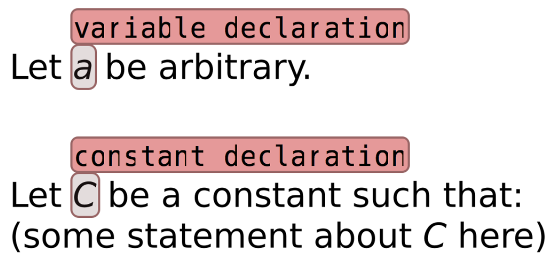
The notation $\mathcal{A}[\mathcal{a} \sim \mathcal{b}]$ is the sentence produced by replacing one or more $\mathcal{a}$'s in $\mathcal{A}$ with $\mathcal{b}$. We write it with the wiggly line to show that the term $\mathcal{b}$ does not need to replace all occurrences of the term $\mathcal{a}$. You can decide which occurrences to replace and which to leave in place. This is not the same as a substitution instance, because $\mathcal{b}$ may replace some or all occurrences of $\mathcal{a}$.

To see the rules in action, consider this proof:

1.	$\forall x, \forall y, x = y$	given
2.	$\exists x, bad(x)$	given
3.	$\forall x, (bad(x) \Rightarrow \neg cute(x))$	given; want $\neg\exists x, cute(x)$
4.	Let E be such that:	
5.	$bad(E)$	∃E 2., 4.
6.	Let f be arbitrary.	
7.	$\forall y, E = y$	∀E 1.
8.	$E = f$	∀E 7.
9.	$bad(f)$	=E 8., 5.
10.	$bad(f) \Rightarrow \neg cute(f)$	∀E 3.
11.	$\neg cute(f)$	⇒E 10., 9.
12.	$\forall x, \neg cute(x)$	∀I 6., 11.
13.	$\neg\exists x, cute(x)$	QN 12.

Quiz Yourself

- If $\mathcal{A}$ stands for the wff $apple(x) \wedge x = y$, then what is $\mathcal{A}[x = t]$?
- Name two wffs of QL that can be concluded using ∀E from the sentence $\forall x, bigger(x, A)$.
- Of the four quantifier rules, which one requires a subproof?
- What is the difference between $\mathcal{A}[x = y]$ and $\mathcal{A}[x \sim y]$?



Let a be arbitrary.

Let C be a constant such that:
(some statement about C here)

Figure 9.1: On top, a variable declaration ME bubble in *Lurch*, as it might appear in a line of one of your proofs. On bottom, a constant declaration ME bubble, which would be followed by a statement about the constant.

9.3 QL Proofs in *Lurch*

You already know how to do proofs in *Lurch* from Chapter 6. In some sense, all this chapter did was add rules to your arsenal. But they have a few new subtleties that the old rules didn't have, so it's worth explaining how those subtleties will play out in your use of *Lurch*.

First, note that *Lurch* is more flexible than the somewhat restrictive QL language defined in this textbook. Herein, we require variables to be lower-case, constants to be upper-case, and predicates to be words. In *Lurch*, however, these are all interchangeable. So no matter whether you wrap a, A, or *apple* in a Meaningful Expression bubble, *Lurch* will put the tag 'variable' at the top of the bubble in each case.

You've learned two new types of proof lines in this chapter, one for declaring variables, 'Let x be arbitrary,' and one for declaring constants to have certain properties, 'Let C be such that…' You can enter such proof lines in *Lurch* as follows.

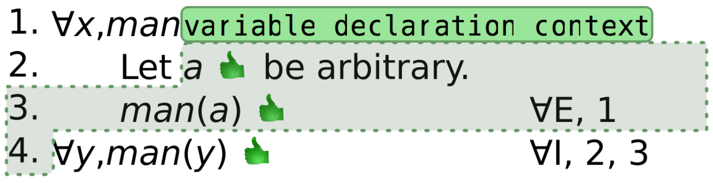
Type the entire line into the software, then place a Meaningful Expression (ME) bubble around only the variable or constant being declared. Click the tag on that bubble and change its type to either 'variable declaration' or 'constant declaration,' as appropriate. The result should be as shown on either the top or bottom of Figure 9.1.

In fact, the words surrounding the ME bubble in such lines are completely optional. By convention, we insert them to make the meaning clear to a human reader. But *Lurch* does not care if those extra words in the proof line are present or not, or whether they are spelled correctly, or whether they are even in English.

Declaring a variable in *Lurch* automatically creates a context bubble. We saw automatically-created context bubbles in SL proofs when we created a subproof. They happen in QL proofs when you either create a subproof or declare a variable. This is consistent with the fact that both of these proof structures are indented in the formal QL system introduced in this chapter. Context bubbles *Lurch* automatically creates for subproofs have the tag 'subproof context,' while those created for variable declarations have the tag 'variable declaration context.' An example appears in Figure 9.2.

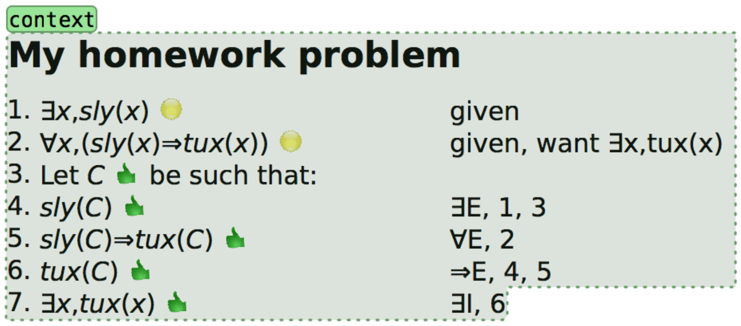
Unlike variable declarations, constant declarations do not create contexts automatically. This is so that when a user writes a document in *Lurch* where they wish to declare several constants at the beginning, those constants will still be usable throughout the document. For example, a mathematical document written in *Lurch* may declare constants such as π and e at the top of the document and assign properties to them, and expect that they will be usable in any proof in the entire document.

This presents a problem if you choose to put several homework exercises in one *Lurch* document, because more than one of them may use the same variable. For example, if your first homework exercise declares the constant K, then later on in your fourth homework exercise, what if you try to declare the constant K



1. ∀x,man
2. Let a be arbitrary.
3. $man(a)$ ∀E, 1
4. $∀y,man(y)$ ∀I, 2, 3

Figure 9.2: An example of an automatically-created variable declaration context in *Lurch*. It begins at the opening of the variable declaration ME bubble (not currently visible) and ends with the statement that uses the variable declaration, ending its scope.



My homework problem

1. ∃x,*sly*(x)	given
2. ∀x,(*sly*(x)⇒*tux*(x))	given, want ∃x,tux(x)
3. Let *C* be such that:	
4. *sly*(*C*)	∃E, 1, 3
5. *sly*(*C*)⇒*tux*(*C*)	∀E, 2
6. *tux*(*C*)	⇒E, 4, 5
7. ∃x,*tux*(x)	∃I, 6

Figure 9.3: A manually inserted context bubble in *Lurch*, to wrap a proof that declares a context, so that other proofs can use the same constant without a conflict.

again? *Lurch* will declare it invalid, because you've already declared K, as if *Lurch* thinks that you're doing one giant proof instead of several! Fortunately, there is a way to fix this.

You can manually insert contexts to separate problems from one another, so that *Lurch* sees separate proofs as exactly that. To do so, simply select an entire proof and click the green button on the toolbar that inserts a context bubble. The result should look like what you see in Figure 9.3. Then *Lurch* knows that this proof is separate from any other work in your document. The constant declared therein will not be usable outside it, and another proof can safely declare the same constant without *Lurch* grading such a declaration incorrect.

In summary, the following tips should be all you need to be able to do the homework assignments from this chapter in *Lurch*.

1. You should respect the QL conventions for capitalization of variables, constants, and predicates, even though *Lurch* will not require you to do so.

2. You can change the type of an ME bubble to a variable or constant declaration by clicking its bubble tag.

3. Declaring variables automatically creates a context for you, but declaring a constant does not.

4. Since you probably only want your constant declarations to persist until the end of your proof, you should wrap each homework problem in a separate context bubble manually.

	YES	NO
Is $\mathcal{A}$ a tautology?	prove $\vdash \mathcal{A}$	give a model in which $\mathcal{A}$ is false
Is $\mathcal{A}$ a contradiction?	prove $\vdash \neg\mathcal{A}$	give a model in which $\mathcal{A}$ is true
Is $\mathcal{A}$ contingent?	give a model in which $\mathcal{A}$ is true and another in which $\mathcal{A}$ is false	prove $\vdash \mathcal{A}$ or $\vdash \neg\mathcal{A}$
Are $\mathcal{A}$ and $\mathcal{B}$ equivalent?	prove $\mathcal{A} \vdash \mathcal{B}$ and $\mathcal{B} \vdash \mathcal{A}$	give a model in which $\mathcal{A}$ and $\mathcal{B}$ 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 $\mathcal{B}$ and $\neg\mathcal{B}$
Is the argument '$\mathcal{P}$, $\therefore$ $\mathcal{C}$' valid?	prove $\mathcal{P} \vdash \mathcal{C}$	give a model in which $\mathcal{P}$ is true and $\mathcal{C}$ is false

Table 9.1: 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.

9.4 Proof-theoretic concepts

Just as in SL, we will use the turnstile symbol '$\vdash$' to indicate that a proof is possible. We write $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\} \vdash \mathcal{B}$ or $\mathcal{A} \vdash \mathcal{B}$ or $\vdash \mathcal{C}$ with the same meanings as in the SL case, except now speaking about the existence of proofs that use QL rules. And $\mathcal{A} \vdash \mathcal{B}$ is still read as '$\mathcal{B}$ is derivable from $\mathcal{A}$.' And the following three definitions are identical to the SL case, and are repeated here for convenience.

A THEOREM is a sentence that is derivable without any premises; i.e., $\mathcal{T}$ is a theorem if and only if $\vdash \mathcal{T}$.

Two sentences $\mathcal{A}$ and $\mathcal{B}$ are PROVABLY EQUIVALENT if and only if each can be derived from the other; i.e., $\mathcal{A} \vdash \mathcal{B}$ and $\mathcal{B} \vdash \mathcal{A}$.

The set of sentences $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$ is PROVABLY INCONSISTENT if and only if a contradiction is derivable from it; i.e., for some sentence $\mathcal{B}$, $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\} \vdash \mathcal{B}$ and $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\} \vdash \neg\mathcal{B}$.

As in SL, a sentence is a theorem if and only if it is a tautology. If we provide a proof of $\vdash \mathcal{A}$ and thus show that it is a theorem, it follows that $\mathcal{A}$ is a tautology; i.e., $\models \mathcal{A}$. Similarly, if we construct a model in which $\mathcal{A}$ is false and thus show that it is not a tautology, it follows that $\mathcal{A}$ is not a theorem.

Just as in Chapter 5, $\mathcal{A} \vdash \mathcal{B}$ if and only if $\mathcal{A} \models \mathcal{B}$. 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 9.1 summarizes when it is best to give proofs and when it is best to give models; it is an updated version of Table 6.1.

In this way, proofs and models give us a versatile toolkit for working with arguments. If we can translate an argument into QL, 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.

9.5 Soundness and completeness

This toolkit is incredibly convenient. It is also intuitive, because it seems natural that provability and semantic entailment should agree. Yet, do not be fooled by the similarity of the symbols '$\models$' and '$\vdash$.' The fact that these two are really interchangeable is not a simple thing to prove.

Why should we think that an argument that *can be proven* is necessarily a *valid* argument? That is, why think that $\mathcal{A} \vdash \mathcal{B}$ implies $\mathcal{A} \models \mathcal{B}$?

This is the problem of SOUNDNESS. A proof system is SOUND if there are no proofs of invalid arguments. Demonstrating that the proof system is sound would require showing that *any* possible proof is the proof of a valid argument. It would not be enough simply to succeed when trying to prove many valid arguments and to fail when trying to prove invalid ones.

Fortunately, there is a way of approaching this in a step-wise fashion. If using the $\wedge$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 $\wedge$E and $\vee$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 $\wedge$I rule. Suppose we use it to add $\mathcal{A} \wedge \mathcal{B}$ to a valid argument. In order for the rule to apply, $\mathcal{A}$ and $\mathcal{B}$ must already be available in the proof. Since the argument so far is valid, $\mathcal{A}$ and $\mathcal{B}$ 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 $\mathcal{A}$ and $\mathcal{B}$ are true. According to the definition of TRUTH IN QL, this means that $\mathcal{A} \wedge \mathcal{B}$ is also true in such a model. Therefore, $\mathcal{A} \wedge \mathcal{B}$ validly follows from the premises. This means that using the $\wedge$E 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. This tedious exercise falls beyond the scope of this book.

Given a proof that the proof system is sound, it follows that every theorem is a tautology.

It is still possible to ask: Why think that *every* valid argument is an argument that can be proven? That is, why think that $\mathcal{A} \models \mathcal{B}$ implies $\mathcal{A} \vdash \mathcal{B}$?

This is the problem of COMPLETENESS. A proof system is COMPLETE if there is a proof of every valid argument. Completeness for a language like QL was first proven by Kurt Gödel in 1929. The proof is beyond the scope of this book.

The important point is that, happily, the proof system for QL is both sound and complete. This is not the case for all proof systems and all formal languages. Because it is true of QL, we can choose to give proofs or construct models— whichever is easier for the task at hand.

Summary of definitions

▷ A sentence $\mathcal{A}$ is a THEOREM if and only if $\vdash \mathcal{A}$.

▷ Two sentences $\mathcal{A}$ and $\mathcal{B}$ are PROVABLY EQUIVALENT if and only if $\mathcal{A} \vdash \mathcal{B}$ and $\mathcal{B} \vdash \mathcal{A}$.

▷ $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$ is PROVABLY INCONSISTENT if and only if, for some sentence $\mathcal{B}$, $\{\mathcal{A}_1, \mathcal{A}_2, \ldots\} \vdash (\mathcal{B} \wedge \neg\mathcal{B})$.

Quiz Yourself

- How do you change an ME bubble into a variable or constant declaration in *Lurch*?
- Why might you want to introduce a context (green bubble) into a *Lurch* document?
- Have the meanings of $\vdash$ and $\vDash$ changed at all from SL to QL?
- Explain what soundness and completeness are, and how they are related.

Practice Exercises

Feel free to use *Lurch* to get immediate feedback on any of the proofs you are asked to do (or complete) below. It will help you avoid developing incorrect habits, and will let you know before you turn your homework in to your instructor whether you will get full credit.

⋆ **Part A** Provide a justification (rule and line numbers) for each line of proof that requires one.

1.	$\forall x, \exists y, (related(x,y) \vee related(y,x))$	given
2.	$\forall x, \neg related(M,x)$	given
3.	$\exists y, (related(M,y) \vee related(y,M))$	
4.	Let A be such that:	
5.	$related(M,A) \vee related(A,M)$	
6.	$\neg related(M,A)$	
7.	$related(A,M)$	
8.	$\exists x, related(x,M)$	

1.	$\forall x, (\exists y, less(x,y) \Rightarrow \forall z, less(z,x))$	given
2.	$less(A,B)$	given
3.	$\exists y, less(A,y) \Rightarrow \forall z, less(z,A)$	
4.	$\exists y, less(A,y)$	
5.	$\forall z, less(z,A)$	
6.	Let c be arbitrary.	
7.	$less(c,A)$	
8.	$\exists y, less(c,y) \Rightarrow \forall z, less(z,c)$	
9.	$\exists y, less(c,y)$	
10.	$\forall z, less(z,c)$	
11.	$less(c,c)$	
12.	$\forall x, less(x,x)$	

1.	$\forall x, (jog(x) \Rightarrow knit(x))$	given
2.	$\exists x, \forall y, less(x, y)$	given
3.	$\forall x, jog(x)$	given
4.	Let A be such that:	
5.	$\forall y, less(A, y)$	
6.	$less(A, A)$	
7.	$jog(A)$	
8.	$jog(A) \Rightarrow knit(A)$	
9.	$knit(A)$	
10.	$knit(A) \wedge less(A, A)$	
11.	$\exists x, (knit(x) \wedge less(x, x))$	

1.	$\neg(\exists x, man(x) \vee \forall x, \neg man(x))$
2.	$\neg\exists x, man(x) \wedge \neg\forall x, \neg man(x)$
3.	$\neg\exists x, man(x)$
4.	$\forall x, \neg man(x)$
5.	$\neg\forall x, \neg man(x)$
6.	$\exists x, man(x) \vee \forall x, \neg man(x)$

⋆ **Part B** Provide a proof of each claim.

1. $\vdash \forall x, fly(x) \vee \neg\forall x, fly(x)$
2. $\{\forall x, (man(x) \Leftrightarrow nice(x)),\quad man(A) \wedge \exists x, related(x, A)\} \vdash \exists x, nice(x)$
3. $\{\forall x, (\neg man(x) \vee less(J, x)),\quad \forall x, (bad(x) \Rightarrow less(J, x)),\quad \forall x, (man(x) \vee bad(x))\} \vdash \forall x, less(J, x)$
4. $\forall x, (cute(x) \wedge dog(T)) \vdash \forall x, cute(x) \wedge dog(T)$
5. $\exists x, (cute(x) \vee dog(T)) \vdash \exists x, cute(x) \vee dog(T)$

Part C Provide a proof of the argument about Billy on p. 88.

Part D Look back at Part B on p. 97. Provide proofs to show that each of the argument forms is valid in QL.

Part E Aristotle and his successors identified other syllogistic forms. Symbolize each of the following argument forms in QL. (Since A, B, and C are predicates in the language Aristotle used, but in QL they are constants, you should replace each with something like *apple*, *bear*, *cabbage*, or perhaps *ay*, *bee*, *cee*, as you prefer.) Add the additional assumptions 'There is an A' and 'There is a B.' Then prove that the supplemented arguments forms are valid in QL.

Darapti: All As are Bs. All As are Cs. ∴ Some B is C.

Felapton: No Bs are Cs. All As are Bs. ∴ Some A is not C.

Barbari: All Bs are Cs. All As are Bs. ∴ Some A is C.

Camestros: All Cs are Bs. No As are Bs. ∴ Some A is not C.

Celaront: No Bs are Cs. All As are Bs. ∴ Some A is not C.

Cesaro: No Cs are Bs. All As are Bs. ∴ Some A is not C.

Fapesmo: All Bs are Cs. No As are Bs. ∴ Some C is not A.

Part F Provide a proof of each claim.

1. $\forall x, \forall y, greater(x,y) \vdash \exists x, greater(x,x)$
2. $\forall x, \forall y, (gets(x,y) \Rightarrow gets(y,x)) \vdash \forall x, \forall y, (gets(x,y) \Leftrightarrow gets(y,x))$
3. $\{\forall x, (angry(x) \Rightarrow bad(x)), \quad \exists x, angry(x)\} \vdash \exists x, bad(x)$
4. $\{nice(A) \Rightarrow \forall x, (man(x) \Leftrightarrow man(A)), \quad man(A), \quad \neg man(B)\} \vdash \neg nice(A)$
5. $\vdash \forall z, (pred(z) \vee \neg pred(z))$
6. $\vdash \forall x, related(x,x) \Rightarrow \exists x, \exists y, related(x,y)$
7. $\vdash \forall y, \exists x, (quiet(y) \Rightarrow quiet(x))$

Part G Show that each pair of sentences is provably equivalent.

1. $\forall x, (angry(x) \Rightarrow \neg bad(x)), \quad \neg\exists x, (angry(x) \wedge bad(x))$
2. $\forall x, (\neg angry(x) \Rightarrow bad(D)), \quad \forall x, angry(x) \vee bad(D)$
3. $\exists x, polite(x) \Rightarrow quiet(C), \quad \forall x, (polite(x) \Rightarrow quiet(C))$
4. $related(C,A) \Leftrightarrow \forall x, related(x,A), \quad \forall x, (related(C,A) \Leftrightarrow related(x,A))$

Part H Show that each of the following is provably inconsistent.

1. $\{sly(A) \Rightarrow tidy(M), \quad tidy(M) \Rightarrow sly(A), \quad tidy(M) \wedge \neg sly(A)\}$
2. $\{\neg\exists x, related(x,A), \quad \forall x, \forall y, related(y,x)\}$
3. $\{\neg\exists x, \exists y, less(x,y), \quad less(A,A)\}$
4. $\{\forall x, (polite(x) \Rightarrow quiet(x)), \quad \forall z, (polite(z) \Rightarrow rowdy(z)), \quad \forall y, polite(y), \quad \neg quiet(A) \wedge \neg rowdy(B)\}$

$\star$ **Part I** Write a symbolization key for the following argument, translate it, and prove it:

> There is someone who likes everyone who likes everyone that he likes. Therefore, there is someone who likes himself.

Part J Provide a proof of each claim.

1. $\{pred(A) \vee qued(B), \quad qued(B) \Rightarrow B = C, \quad \neg pred(A)\} \vdash qued(C)$
2. $\{M = N \vee N = O, \quad apple(N)\} \vdash apple(M) \vee apple(O)$
3. $\{\forall x, x = M, \quad related(M,A)\} \vdash \exists x, related(x,x)$
4. $\neg\exists x, x \neq M \vdash \forall x, \forall y, (pretty(x) \Rightarrow pretty(y))$
5. $\forall x, \forall y, (related(x,y) \Rightarrow x = y) \vdash related(A,B) \Rightarrow related(B,A)$
6. $\{\exists x, junk(x), \quad \exists x, \neg junk(x)\} \vdash \exists x, \exists y, x \neq y$
7. $\{\forall x, (x = N \Leftrightarrow mad(x)), \quad \forall x, (old(x) \vee \neg mad(x))\} \vdash old(N)$
8. $\{\exists x, dog(x), \quad \forall x, (x = P \Leftrightarrow dog(x))\} \vdash dog(P)$
9. $\{\exists x, \big(kind(x) \wedge \forall y, (kind(y) \Rightarrow x = y) \wedge big(x)\big), \quad kind(D)\} \vdash big(D)$
10. $\vdash pretty(A) \Rightarrow \forall x, (pretty(x) \vee x \neq A)$

Part K Without using the QN rule, prove $\neg\exists x, \neg\mathcal{A} \vdash \forall x, \mathcal{A}$

Part L Look back at Part D on p. 98. For each argument: If it is valid in QL, give a proof. If it is invalid, construct a model to show that it is invalid.

$\star$ **Part M** 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. $\forall x, por(x) \Rightarrow que(C), \quad \forall x, (por(x) \Rightarrow que(C))$
2. $\forall x, por(x) \wedge que(C), \quad \forall x, (por(x) \wedge que(C))$

3. $qrs(C) \lor \exists x, qrs(x), \quad \exists x, (qrs(C) \lor qrs(x))$
4. $\forall x, \forall y, \forall z, between(x, y, z), \quad \forall x, between(x, x, x)$
5. $\forall x, \forall y, divisible(x, y), \quad \forall y, \forall x, divisible(x, y)$
6. $\exists x, \forall y, divisible(x, y), \quad \forall y, \exists x, divisible(x, y)$

$\star$ **Part N** 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. $\forall x, \exists y, related(x, y), \quad \therefore \exists y, \forall x, related(x, y)$
2. $\exists y, \forall x, related(x, y), \quad \therefore \forall x, \exists y, related(x, y)$
3. $\exists x, (pred(x) \land \neg qued(x)), \quad \therefore \forall x, (pred(x) \Rightarrow \neg qued(x))$
4. $\forall x, (small(x) \Rightarrow tiny(A)), \quad small(D), \quad \therefore tiny(A)$
5. $\forall x, (angry(x) \Rightarrow bad(x)), \quad \forall x, (bad(x) \Rightarrow cruel(x)), \quad \therefore \forall x, (angry(x) \Rightarrow cruel(x))$
6. $\exists x, (dog(x) \lor emu(x)), \quad \forall x, (dog(x) \Rightarrow fox(x)), \quad \therefore \exists x, (dog(x) \land fox(x))$
7. $\forall x, \forall y, (related(x, y) \lor related(y, x)), \quad \therefore related(J, J)$
8. $\exists x, \exists y, (related(x, y) \lor related(y, x)), \quad \therefore related(J, J)$
9. $\forall x, pred(x) \Rightarrow \forall x, qued(x), \quad \exists x, \neg pred(x), \quad \therefore \exists x, \neg qued(x)$
10. $\exists x, man(x) \Rightarrow \exists x, nice(x), \quad \neg \exists x, nice(x), \quad \therefore \forall x, \neg man(x)$

Chapter 10

Real numbers

10.1 An important transition

This book has introduced two proof systems. We built QL on top of SL, and saw in Chapter 9 that QL is reliable, in the sense that it is both sound and complete for a natural collection of models. The rest of this book leverages QL to build, from the ground up, some mathematics with which you're probably already familiar.

Historically speaking, this order of topics is backwards. Humanity investigated questions of mathematics for a long time without ever formalizing the logic they used to reason about mathematics. Logic as a formal part of mathematics only really got started in the 20^{th} century. But the book has proceeded in this order because it makes sense to learn the tools before we use them! We now begin to apply the logic we have learned to proving mathematical facts.

The goal in all of this is to see the value, for mathematics, of all the logic that you have learned so far in this book. We will be proving, using ironclad logical systems, a vast array of mathematical facts, beginning with very small and simple facts, and ending with facts from calculus. You will see that all you've learned throughout your past mathematical education is actually supported by airtight mathematical arguments. The rules of algebra and calculus are what they are because logic demands it.

Be sure to take some satisfaction from mastering both the ability to understand proofs that support advanced mathematics *and* the ability to write those very proofs as well. It is a skill and a viewpoint that a very small percentage of people get to experience.

From here onward, our proofs will unavoidably be more complicated and lengthy than before. To accomplish our goal (building a reliable foundation of mathematics using logic) a lot of work must be done! Many of the mathematical facts that we take for granted are surprisingly tricky to justify. To help with this difficulty, we are permitted some new tools.

Each chapter from here on will introduce new shortcuts, each of which allows you to skip some of the most tedious steps in a proof. These shortcuts are not introduced out of sympathy, or out of fear that you would otherwise give up and quit! Rather, professional mathematicians use these same shortcuts all the time in their own writing.

Knowing how to read and write using these shortcuts is an essential part of proofs in mathematics. Without these shortcuts, mathematical proofs would be so long that they would be very difficult to understand. The shortcuts not only help write a proof more easily, but they help its readers better understand the argument.[1]

10.2 Shortcuts about statements

This section introduces some new shortcuts you can now use when writing proofs. Furthermore, proofs written from here on in the textbook will begin to use these same shortcuts, so you need to be ready for them as you read proofs, as well.

These are not the only new shortcuts you'll get; these are the shortcuts that pertain to mathematical statements alone, whether they're in a proof or not. Later sections in this chapter will introduce new shortcuts that relate to writing different kinds of proofs.

It is not necessary to refer to these shortcuts when using them in your work. You can simply use them silently, and the text will do the same.

Expanded language

We now begin using, in our logical language, many of the common symbols from mathematics. Here are a few examples, but we are not limited only to these examples. Many different mathematical symbols will begin to appear in our work, each coming with corresponding rules that define its meaning. We have not yet seen rules that define the meanings of all the following symbols, but the next section will introduce those rules.

1. In addition to writing relations in the way we have in QL, as in $less(x, y)$ and $greater(a, b)$, we will now be permitted to use the standard mathematical symbols for common relations. For instance, the usual less than, greater than, less than or equal to, and greater than or equal to relationships in mathematics will be written as $x < y$, $x > y$, $x \leq y$, and $x \geq y$, respectively.

   In the past, equality was the one special relation that we wrote with a symbol between the variables, as in $x = y$ rather than $equal(x, y)$. We now permit all relations that have such a shorthand symbol to use it.

   Example statements in this new language:

   $$\forall x, \exists y,\ x > y \qquad\qquad \exists y,\ y \geq 0$$

2. It is not only relations that we will be adding to our language; we can now use functions and operations as well. In the past, the only "nouns" in our mathematical language were the variables and constants a, b, c, A, B, C, and so on. Now we will be able to add them, as in $a + b$, multiply them, as in $C \cdot D$, and apply functions to them, as in $f(x)$.

   These new ways to express mathematical objects can still be used inside relations as well. For example, we can write $x < f(x)$, or $A \cdot y = B + y$.

   Example statements in this new language:

   $$\forall x,\ x \cdot x > 0 \qquad\qquad \exists y,\ \exists z,\ 3 \cdot g(y) - 5 = z$$

Naturally, as you read these new privileges, you may wonder whether *Lurch* supports them. That is, can *Lurch* understand me if I type in $\exists x,\ 3 \cdot x - 5 = 0$? The answer is yes, but I don't want to focus on that right now. For the majority of this chapter, I'll explain the new privileges only in terms of the logic and mathematics we're doing, ignoring the software that can check it. At the end of the chapter (Section 10.5) I will then review the new shortcuts this chapter has introduced, and explain how each can be done in *Lurch*.

[1]Chapters 10 through 14 were not in P.D. Magnus's original text. They were added by Nathan Carter in 2015.

Rules about mathematics

If we're going to have in our language these mathematical symbols, we must have some way to reason about them. As you know from your study so far in this text, the rules for QL make no assumptions at all about the meanings of any of the relations in the language. We can write the statement $\forall x, \exists y, \mathit{bigger}(x, y)$ and cannot prove it to be true or false, unless we have some further assumptions. We can construct models in which the statement is true, and models in which it is false, *depending on the meaning of the relation "bigger."*

The same is true about all relations in the language, including the new ones we've just introduced, $<$, $>$, $\leq$, and $\geq$. And the same is true about the functions and operations we've introduced into the language; their meaning is not specified by QL alone. Thus we need to adopt some conventions about the meanings of these mathematical operations and relations.

Such conventions are called *axioms.* In this chapter we will list axioms that describe the essential meanings of the various operations and relations we use on real numbers, including $+$, $<$, and the others mentioned above. Each axiom will be a single statement in the language of QL, expanded as described above. Here is an example axiom.

$$\forall a, \forall b, \forall c, (a + b) + c = a + (b + c)$$

It is called the associativity of addition, and it says that the position of parentheses among a sequence of added terms doesn't matter.

All axioms are single statements in our new language. The way you access them in proofs is to treat them as *new rules of inference.* They take no inputs, and produce a single output, the statement of the axiom itself. For example, to use the above axiom in a proof, I would just insert the line $\forall a, \forall b, \forall c, (a+b)+c = a+(b+c)$ into the proof, and the reason would be "associativity of addition."

Speaking less formally

Before we begin listing all the axioms about the operations and relations on real numbers, it will be helpful to adopt a few shortcuts that will help us speak less like logicians and more like mathematicians. After all, this is the chapter in which we transition from studying logic alone to seeing how logic applies to mathematics.

As in the example above, axioms almost always begin with universal quantifiers. They tend to make claims that are true across the whole universe of discourse, and thus begin with universal quantifiers for most (if not all) of their variables. Consequently, we adopt the shortcut that you are permitted to omit those initial universal quantifiers. For example, the above axiom could be stated more simply as follows.

$$(a + b) + c = a + (b + c)$$

The reader is forced to realize that the three variables in that statement appear free, and so the reader concludes that they are *implicitly* universally quantified.

The second convention we will adopt is simpler and easier to deal with, as both a reader and a writer. The convention in mathematics is to almost never use the logical symbols for not, and, or, and if-then. Rather, these words are simply written in ordinary English sentences.

For example, if we wish to state the axiom for real numbers that $a \neq 0 \Rightarrow a \cdot \frac{1}{a} = 1$, then we would instead write it as an English sentence, "if $a \neq 0$ then $a \cdot \frac{1}{a} = 1$."

10.3 Axioms

With these shortcuts in mind, we can now state the 17 axioms for the real numbers. Keep in mind that each of these can be treated as a new rule that you can use in your proofs. When used as a rule, an axiom requires no inputs, and produces just itself as an output.

These 17 (very basic!) facts about the real numbers are (almost) all we need to know to prove countless other facts about the real numbers, including some very complex facts about calculus! That will be our business for the next few chapters of this text.

Axiom 1 (associativity of $+$)**.** *For any real numbers a, b, c, we have $(a+b)+c = a+(b+c)$.*

Axiom 2 (commutativity of $+$)**.** *For any real numbers a, b, we have $a+b = b+a$.*

Axiom 3 (additive identity)**.** *For any real number a, $a+0 = a$.*

Axiom 4 (additive inverse)**.** *Every real number a has a negative, written $-a$, and $a + -a = 0$.*

Axiom 5 (associativity of $\cdot$)**.** *For any real numbers a, b, c, we have $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.*

Axiom 6 (commutativity of $\cdot$)**.** *For any real numbers a, b, we have $a \cdot b = b \cdot a$.*

Axiom 7 (multiplicative identity)**.** *For any real number a, $a \cdot 1 = a$.*

Axiom 8 (multiplicative inverse)**.** *For any real number a, if $a \neq 0$ then $a \cdot \frac{1}{a} = 1$.*

Axiom 9 (the identities are different)**.** $0 \neq 1$

Axiom 10 (distributivity)**.** *For any real numbers a, b, c, $a \cdot (b+c) = a \cdot b + a \cdot c$.*

Axiom 11 (Archimedean property)**.** *For any real numbers a and b, $a \leq b$ or $b \leq a$.*

Axiom 12 (linear order)**.** *For any real numbers a, b, if $a \leq b$ and $b \leq a$ then $a = b$.*

Axiom 13 (transitivity)**.** *For any real numbers a, b, c, if $a \leq b$ and $b \leq c$ then $a \leq c$.*

Axiom 14 (addition preserves order)**.** *For any real numbers a, b, c, if $a \leq b$ then $a + c \leq b + c$.*

Axiom 15 (multiplication preserves order)**.** *For any real numbers a, b, c, if $a \leq b$ and $0 \leq c$ then $a \cdot c \leq b \cdot c$.*

Axiom 16 (definition of $<$)**.** *We write $a < b$ to mean $a \leq b$ and $a \neq b$.*

Axiom 17 (definition of $\geq$)**.** *We write $a \geq b$ to mean $b \leq a$.*

Axiom 18 (definition of $>$)**.** *We write $a > b$ to mean $b < a$.*

Quiz Yourself

- For the axioms above, what is the implied universe of discourse?
- How is an axiom actually a rule of inference?

10.4 Theorems

Proving basic theorems from the axioms

In mathematics, we usually write a theorem and its proof in a document, one after the other. You know the proof is complete when you see a little box on the right-hand edge of the page. Here is the general format, although without an actual proof.

Theorem. *The statement of the theorem would go here.*

Proof. The proof of the theorem goes here. It may be large, occupying many lines, as our proofs have been in previous chapters. $\square$

As you can see, the previous line ended with a little box all the way on the rightmost edge, so now you know the proof is complete. Although this is irrelevant for short proofs, we will soon be doing long proofs that are written in a non-formal style, and the box will be helpful. For example, look ahead to the proof of Theorem 3 on page 144.

Below is an actual example of a theorem-and-proof in the above format. This theorem is a *very* simple mathematical fact, and its proof is based on the axioms from the previous section.

Theorem 1. $-0 = 0$

Proof.

1.	$\forall a, a + -a = 0$	Axiom 4
2.	$0 + -0 = 0$	∀E 1.
3.	$\forall a, a + 0 = a$	Axiom 3
4.	$-0 + 0 = -0$	∀E 3.
5.	$\forall a, \forall b, a + b = b + a$	Axiom 2
6.	$0 + -0 = -0 + 0$	∀E 5.
7.	$0 + -0 = -0$	=E 4., 6.
8.	$-0 = 0$	=E 2., 7.

$\square$

We've done our first mathematical proof! This may seem a little silly, because all we've proven is that $-0 = 0$, which is hardly a surprise. However, keep in mind that what we're doing is building mathematics up from just 17 axioms. We've now proven a new mathematical fact from only the very small set of assumptions in Section 10.3. We will prove many more theorems from those same axioms, and because we will have solid, logical proofs of each, we will know that our theorems are reliable. Proof is the essence of establishing mathematical truths.

But mathematicians don't write proofs as formally and logically as I did in the example above. They use even more shortcuts!

First, the pattern in the first six lines of the above proof is clear: We cite an axiom to introduce a universally quantified sentence, then use ∀E to apply it to the situation we care about. Thus using an axiom is always a two-step process. Because this is tedious, we permit the shortcut of doing it all in one step. Thus the above proof can be shortened by three lines as follows.

1.	$0 + -0 = 0$	Axiom 4
2.	$-0 + 0 = -0$	Axiom 3
3.	$0 + -0 = -0 + 0$	Axiom 2
4.	$0 + -0 = -0$	=E 2., 3.
5.	$-0 = 0$	=E 1., 4.

Furthermore, mathematicians do not usually mention rules of logic in their proofs. The rule "=E" is a logical term that doesn't mean much to most mathematicians. They would call it "substitution" instead. Furthermore, it's expected that the reader can find out how the substitution was done without requiring line numbers to be cited. Thus the following simplification of the same proof is also permitted.

1.	$0 + -0 = 0$	Axiom 4
2.	$-0 + 0 = -0$	Axiom 3
3.	$0 + -0 = -0 + 0$	Axiom 2
4.	$0 + -0 = -0$	substitution
5.	$-0 = 0$	substitution

Finally, since we aren't actually citing any line numbers, there's no reason to have them any more. As we remove the line numbers, it's common to then turn each line of the proof into a single sentence, and put them into a paragraph. I do so here in an extremely simple way that comes out sounding rather repetitive; we will improve the writing thereafter.

Proof. We know $0+-0 = 0$ from Axiom 4. We know $-0+0 = -0$ from Axiom 3. We know $0+-0 = -0+0$ from Axiom 2. We conclude $0 + -0 = -0$ by substitution. We conclude $-0 = 0$ by substitution. □

This is a little bit boring to read, and sounds like it was created by a computer. So we can try to be a little bit more smooth with our writing, varying the type of sentences, to make the proof easier to read. Furthermore, we can cite the axioms by name rather than number, so that the reader doesn't need to look the axiom up to see what it says; the name can remind them. The final result is the following proof that any mathematician would accept.

Proof. The additive inverse field axiom tells us that $0 + -0 = 0$. And the additive identity axiom tells us that $-0 + 0 = -0$. By the commutativity of addition, we know that $-0 + 0 = 0 + -0$, and so substituting once yields $0 + -0 = -0$, and substituting again gets us what we want, $-0 = 0$. □

This may seem far from an iron-clad proof! It may seem very vague and imprecise, since you have been trained so far in formal logic. However, here is the key question to ask yourself when reading an informal proof like the one above: Is that proof a sufficient hint to outline every step I would need to construct a corresponding formal proof? In this case, it is; just as we took very simple steps to reduce a formal proof down to this short form, you could take those same steps in reverse to reconstruct the formal proof.

In any proof you encounter that's written in this formal style, if you find that it's a sufficiently detailed hint that you could write a formal proof from it, then you have good reason to accept the informal proof as correct justification of the mathematical fact it claims to prove.

However, if you find an informal proof that does *not* serve as a sufficiently detailed outline for constructing a corresponding formal proof, then you can call into question whether the author of the informal proof has really given a complete proof. You can even point to the specific sentence or phrase in the informal proof that cannot be correctly converted back to a formal proof, and cite that sentence as the source of the trouble.

As you write informal proofs, you can come at them from either of two directions. You can either construct a formal proof first, then simplify it down to an informal version, as I have done in this section. Then, as you get more familiar with informal proofs, you can begin writing them informally without first constructing a formal version. When you advance to writing informal proofs in that way, be sure to always check your informal proof against the standard just mentioned: Does it contain enough information for a typical reader to use it as a guide for easily reconstructing a formal proof?

You can now try proving Theorems 4 through 6 in the Practice Exercises for this chapter. The remaining sections in the chapter teach additional shortcuts, but you will not need them for those first few problems.

Equation-based proofs

Because many of the axioms from Section 10.3 involve equations, most of the theorems we will prove from them (and the steps in proving those theorems) will also involve equations. On a related note, the reason "substitution" (shorthand for =E) will therefore be used frequently as well. You are probably already familiar with proofs that work this way.

For example, if you've taken a high-school algebra class, and your instructor asked you to simplify a tangled bit of algebra, you might have been required to show the step-by-step work of such a simplification something like the following.

$$\begin{aligned}\frac{x^2-8x-9}{x+1}-(5-y)^2 &= \frac{(x+1)\cancel{(x-9)}}{\cancel{x+1}}-(5-y)^2\\ &= x+1-(5-y)^2\\ &= x+1-(25-10y+y^2)\\ &= x-24+10y-y^2\end{aligned}$$

The only difference between such work and what's permitted in a mathematical proof is that in a mathematical proof, each step of work requires a reason to justify it.

Consider the following theorem as an example, with its accompanying informal proof.

Theorem 2. *For any a, $0a = 0$.*

Proof.

$$\begin{aligned} 0 &= 0a+(-(0a)) && \text{additive inverse}\\ &= (0+(-0))a+(-(0a)) && \text{additive inverse}\\ &= (0a+(-0)a)+(-(0a)) && \text{distributivity}\\ &= (0a+0a)+(-(0a)) && \text{Theorem 1}\\ &= 0a+(0a+(-(0a))) && \text{associativity}\\ &= 0a+0 && \text{additive inverse}\\ &= 0a && \text{additive identity}\end{aligned}$$

□

There are two shortcuts used in this proof that we have not seen before. I call your attention to them here, not only so that you can understand the above proof, but also so that you are prepared to use them in your own work later.

First, notice that it is permitted to cite previously proven theorems. The proof above cites Theorem 1 to justify its fourth step. We use them in exactly the same way we used axioms. Specifically, we can skip the $\forall$E step, and just apply the theorem to the particular values useful in our proof.

Second, the entire theorem is a sequence of equations strung together in a chain. This could easily be converted into a formal proof by turning each link in the chain into a separate equation in a line-by-line proof, and applying substitution (=E) as often as needed (usually once per line). As an example, the first few lines of the proof above could be made more formal as follows.

$0 = 0a + (-(0a))$	additive inverse
$0 + (-0) = 0$	additive inverse
$0 = (0 + (-0))a + (-(0a))$	substitution
$(0 + (-0))a = 0a + (-0)a$	distributivity
$0 = 0a + (-0)a + (-(0a))$	substitution
$-0 = 0$	Theorem 1
$0 = 0a + 0a + (-(0a))$	substitution

As you read proofs that are written in the equation-chain style of the proof of Theorem 2, you must be ready for the fact that substitution has been silently applied, and you'll need to spot where! If you find that the only missing step is substitution, then the equation chain is an acceptable shortcut for a more formal proof. If additional steps are missing, then the equation chain is skipping too much, and should be called into question.

Using equation chains where necessary, consider trying some of the next theorems in the exercises, such as Theorems 7 through 13. (Only some of those proofs are made easier using equation chains; consider carefully when it is useful and when it is not.)

When you eventually move beyond *Lurch*, you may write proofs on paper, or type them in software such as Microsoft Word. In Word, you can create the equation chain structure by inserting a matrix and arranging the statements and reasons in it. You will want to left-aligne many of the columns, which you can do by right-clicking a column and choosing "Column Alignment."

Making longer proofs more tractable

As you proceed to the proofs of the later theorems in the Practice Exercises section of this chapter, the proofs will get longer. It will therefore be helpful to have even more shortcuts. I conclude this chapter by stating a final pair of shortcuts, and giving an example of their use.

1. You do not need to specify every single step of logic, especially those that you feel would be easiest for the reader to fill in, and most boring for you to include. But *always* be sure to give enough breadcrumbs that the reader could easily create the full logical formal proof if needed.
2. Mathematicians do not usually reference the rules of logic by name. Thus if you need to apply a rule of SL or QL, you can apply it, and omit the reason. It is understood that at this point in the course, you have mastered the rules of SL and QL enough to recognize them even if their names are not explicitly cited.

Here is an example proof that uses some of these shortcuts. See if you can spot where they are used.

Theorem 3. *For any a, b, if $a \cdot b = 0$ then either $a = 0$ or $b = 0$.*

Proof. Let a, b be arbitrary. Assume $a \cdot b = 0$ and $a \neq 0$. Then there is a real number $\frac{1}{a}$ such that $a \cdot \frac{1}{a} = 1$ (by the multiplicative inverse axiom).

$$\begin{aligned}
b &= 1 \cdot b && \text{multiplicative identity} \\
&= \left(\frac{1}{a} \cdot a\right) \cdot b && \text{statement above} \\
&= \frac{1}{a} \cdot (a \cdot b) && \text{associativity} \\
&= \frac{1}{a} \cdot 0 && \text{assumption from above} \\
&= 0 && \text{Theorem 2}
\end{aligned}$$

So if $a \neq 0$ then $b = 0$, meaning that either $a = 0$ or $b = 0$. □

Did you notice when these two new shortcuts were used? Compare your answer with the following: In the second and fourth lines of the equation chain, the R rule from SL was used without stating its name. In the final sentence of the proof, the MC rule was applied silently.

Quiz Yourself

- When reading a proof written in an informal style, to what standard of correctness should you hold it?
- In an equation-based proof, what rule of logic is often used but rarely explicitly stated?

10.5 Shortcuts in *Lurch*

Most of the shortcuts in this chapter can also be used in *Lurch*, should you wish to do so. This is the final chapter in which *Lurch* supports the proofs in the text; hereafter, we become advanced enough mathematically that we will want to use language and shortcuts that *Lurch* cannot yet support.

The most important step in order to have *Lurch* understand the mathematics introduced in this chapter is to choose the correct topic. Looking back to Figure 6.1 on page 64, you may remember the first time you did this for the language and rules of SL. Then when you graduated to the language and rules of QL, you needed a new topic. Now, we need a third topic.

1. From the *Lurch* file menu, click "Choose topic..."
2. On the list that appears, choose "forallx in Lurch," then the subtopic "Real numbers," and finally the item called "Blank document."
3. Click the checkbox next to "Show this topic's contents in every new Lurch window."
4. Click OK.

You can now type mathematical expressions that involve the symbols $>$, $<$, $\geq$, $\leq$, $\cdot$, $+$, $-$, and expressions like $f(x)$. See Figure 10.1 for two examples.

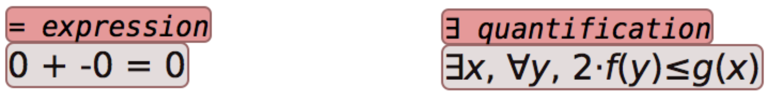
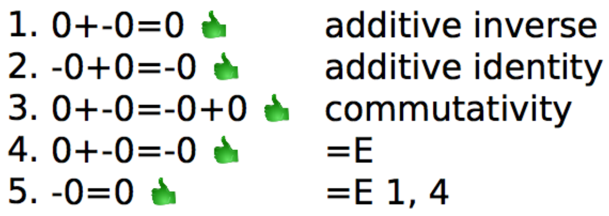


Figure 10.1: Two *Lurch* meaningful expressions in the mathematical language introduced in this chapter



1. 0+-0=0 👍	additive inverse
2. -0+0=-0 👍	additive identity
3. 0+-0=-0+0 👍	commutativity
4. 0+-0=-0 👍	=E
5. -0=0 👍	=E 1, 4

Figure 10.2: The proof of Theorem 1 from the text, done in *Lurch*

Naturally, you can use the axioms introduced in Section 10.3 in *Lurch* as well. There are two things you need to know.

First, as on page 141, you should not bother introducing the axiom in its universally quantified form, and then applying the ∀E rule. Rather, the axioms in *Lurch* have been designed to be used without that step.

Second, the axioms are not numbered, as they are in Section 10.3; rather, they use the *names* from that section instead. (Thus later editions of this text may renumber the axioms without *Lurch* going out-of-date.) So instead of citing "Axiom 2," you would cite "commutativity" or "commutativity of addition." As always, in *Lurch*, the list of rules to which you have access can be found on the Meaning menu, under the item "List all defined rules."

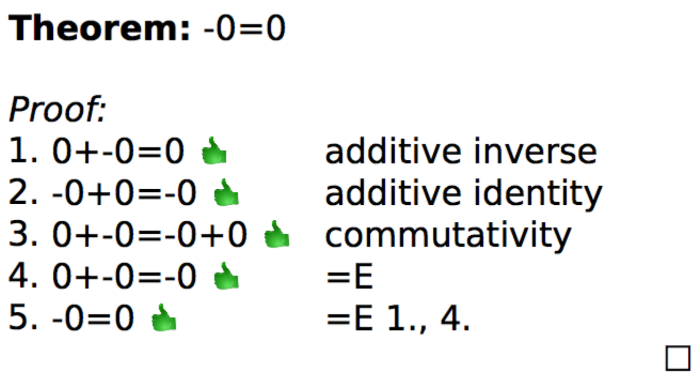
An example proof in *Lurch* of Theorem 1 from the text appears in Figure 10.2. Although no bubbles are shown, there are naturally meaningful expression bubbles around each statement and reason bubbles around each reason.

You will notice in that figure that another shortcut is permissible in *Lurch*. Most of the lines do not cite line numbers, and yet *Lurch* judges them correct anyway. The rule for when you can omit premise citations in *Lurch* is very simple: If the premises you need immediately precede the conclusion you're writing, then you don't need to cite them; *Lurch* will find them automatically. Thus in Figure 10.2, line 4 did not need to cite any premises, because the appropriate ones to cite are the immediately preceding two lines, 2 and 3. But line 5 did need to cite premises, because one of them was line 1, which is not immediately before line 5. Technically, I could have just cited line 1, and left *Lurch* to discover line 4, and that would also have been permissible.

You will notice, however, that the substitution rule is still called =E in *Lurch*. This is just because the word "substitution" was not used when defining the rule set. If you really want to use the word substitution, feel free to open up the document that ships with *Lurch* that defines the rules for SL, and add a new label "substitution" to the =E rule! *Lurch* will learn it, and allow you to use that name thereafter. But by default, it does not know the word "substitution" as a reason.

You can certainly also add some formatting near your proofs to imitate the style introduced in the previous section. For instance, you can state the theorem with the word "Theorem:" in bold before your proof, and insert "Proof:" before the proof and a box after it. The result is shown in Figure 10.3, and involves no new meaningful expressions or other bubbles at all. The new text is simply decoration to help the reader.[2]

[2]The box shown there was inserted with the `\ballotbox` command, but you can also find it on one of the symbol palettes.



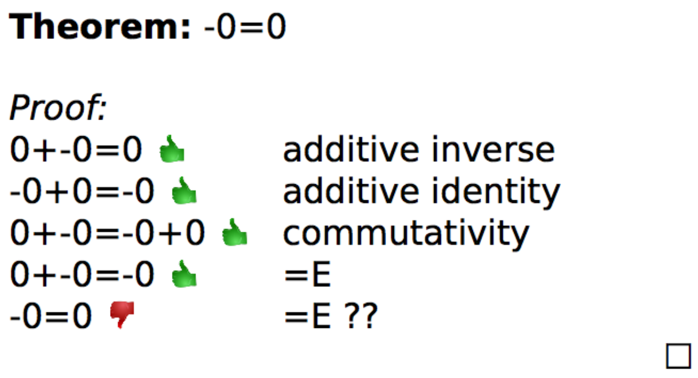
Theorem: -0=0

Proof:

1. 0+-0=0 👍 additive inverse
2. -0+0=-0 👍 additive identity
3. 0+-0=-0+0 👍 commutativity
4. 0+-0=-0 👍 =E
5. -0=0 👍 =E 1., 4.

□

Figure 10.3: The proof from Figure 10.2, with formatting around it for readability



Theorem: -0=0

Proof:

0+-0=0 👍 additive inverse

-0+0=-0 👍 additive identity

0+-0=-0+0 👍 commutativity

0+-0=-0 👍 =E

-0=0 👎 =E ??

□

Figure 10.4: The proof from Figure 10.3, with the line numbers removed. We see that we can no longer cite the first line, since it doesn't have a number. See Section 10.5 for an explanation of how to fix this problem.

If we want to begin converting the proof shown in Figure 10.2 to the sentences-and-paragraphs style taught in the previous section, we will need to remove the line numbers. If we remove the numbered list, we discover a problem, as shown in Figure 10.4.

The solution to this problem is to do what mathematicians do. If there is an important statement, formula, or equation that the reader will need to remember later, it is often marked with a $*$ or similarly noticeable symbol, and then referred to by that mark later. You can add such a label to any meaningful expression in *Lurch*, just as you add reasons or premises. On the Meaning menu, you will find the "Insert label" command, which can be used to make any section of text a label for what comes before it. The top of Figure 10.5 shows how the first line of the proof would need to be changed to insert a label, and the bottom of that same figure shows how the final line of the proof would need to be changed to add a citation of the first line, using that label.

Now the proof is judged correct by *Lurch*, and completely independent of line numbering. We can therefore insert as many other words in between the statements and reasons as we like, in order to create complete sentences, and help the reader understand our work. We do not need to use tabs to line up reasons, nor lines to separate statements. The result ends up looking like Figure 10.6.

Lurch can still understand and give feedback on proofs done in this way, because the bubbles that indicate meaningful expressions, labels, reasons, and premises indicate to *Lurch* the structure of your argument. The

It has been right-aligned using the Format menu.

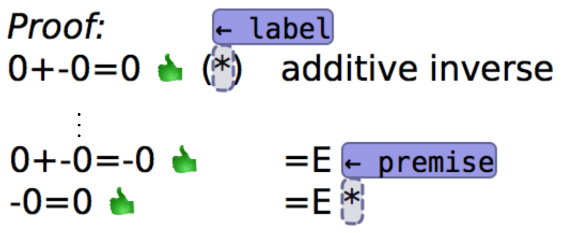
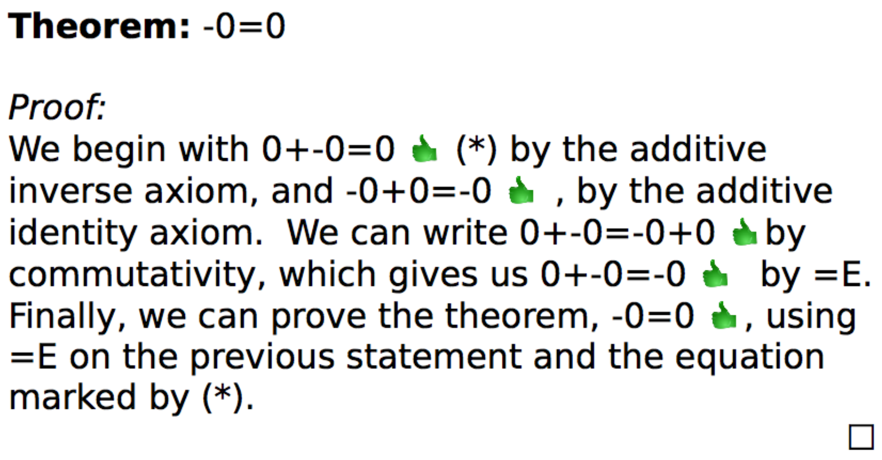

Proof:
0+-0=0 (*) additive inverse
⋮
0+-0=-0 =E premise
-0=0 =E *

Figure 10.5: Introducing a label and a citation thereof, to fix the problem shown in Figure 10.4


Theorem: -0=0

Proof:
We begin with 0+-0=0 (*) by the additive inverse axiom, and -0+0=-0 , by the additive identity axiom. We can write 0+-0=-0+0 by commutativity, which gives us 0+-0=-0 by =E. Finally, we can prove the theorem, -0=0 , using =E on the previous statement and the equation marked by (*).

□

Figure 10.6: A proof in *Lurch* without the formal style

formal style of two columns of statements and reasons is unnecessary.

Lurch provides one final privilege to help you make more natural sentences when writing your proofs in this paragraph-based style. If you wish to place a label, reason, or premise *before* the meaningful expression it modifies, aim your mouse at the arrow in the bubble and click the left or right mouse button. Clicking the left button makes the bubble modify more meaningful expressions to the left; clicking the right button makes it modify more meaningful expressions to the right. By default, a label, reason, or premise bubble modifies only one adjacent meaningful expression, the one to the left. With this flexibility, you can have it modify one or more adjacent meaningful expressions to the right or to the left.

Figure 10.7 shows a snippet from a document in which a reason bubble modifies the meaningful expression that *follows* it. This is clear from the direction that the arrow in the bubble is pointing. Feel free to use this flexibility to make your proofs more readable. The proof from which Figure 10.7 was taken is shown in full in Figure 10.8.

Despite all these new privileges, there are, however, some things that *Lurch* cannot do. This chapter suggested that you may skip the citation of commonly -used rules of logic, or not refer to them directly by name. The topic you will be using still requires that you reference all rules of logic by name. Also, *Lurch*

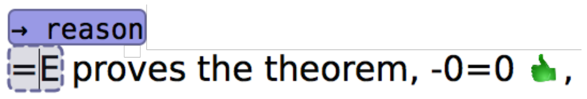
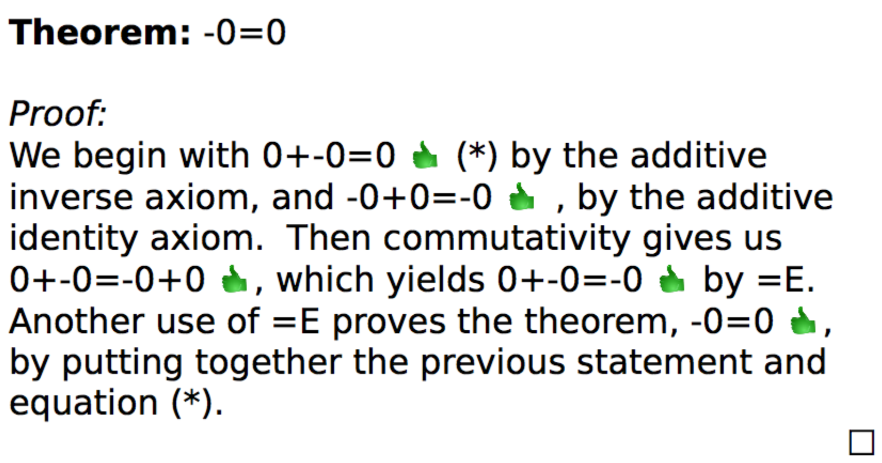

=E proves the theorem, -0=0 ,

Figure 10.7: A reason bubble that modifies the meaningful expression *after* it, as shown by the label in the bubble



Theorem: -0=0

Proof:
We begin with 0+-0=0 👍 (*) by the additive inverse axiom, and -0+0=-0 👍 , by the additive identity axiom. Then commutativity gives us 0+-0=-0+0 👍, which yields 0+-0=-0 👍 by =E. Another use of =E proves the theorem, -0=0 👍, by putting together the previous statement and equation (*).

□

Figure 10.8: A proof in a flexible writing style, leveraging the flexibility shown in Figure 10.7

does not yet support proofs that include chains of equations, like the proof of Theorem 2 from this chapter.

If you wish to transition to writing your proofs in another environment, such as Microsoft Word with Equation Editor, a LaTeX environment, or good ol' pencil and paper, you can leverage those shortcuts. But you will lose the benefit of constant feedback on whether your proofs are correct.

Practice Exercises

Part A

Prove the following three simple theorems using the axioms in Section 10.3. You may choose to give your proofs in a formal style or an informal style, as you prefer.

Theorem 4. $0 + a = a$

Theorem 5. *If $a + c = b + c$ then $a = b$.*

Theorem 6. *If $a \cdot c = b \cdot c$ and $c \neq 0$ then $a = b$.*

Part B

Provide proofs for the following theorems.

Theorem 7. *For any a, b, if $a > 0$ and $b > 0$ then $ab > 0$.*

Theorem 8. *For any a, b, if $a > 0$ and $b < 0$ then $ab < 0$.*

Theorem 9. *For any a, b, $(-a)(-b) = ab$.*

Hint: A chain of equalities will probably be helpful in proving Theorem 9.

Theorem 10. $(-1)(-1) = 1$

Theorem 11. $-(-a) = a$

Part C

Provide proofs for the following theorems.

Theorem 12. *If $a < b$ then $a + c < b + c$.*

Hint: Most likely, a part of your proof of Theorem 12 will use the style of argument labeled "for reductio" earlier in this text. The more common mathematical way to phrase that does not use Latin words. Rather, when we make an assumption for reductio, we say "Assume towards a contraction that A," for whatever statement A we wish to assume. In your write-up of this proof, use that phrasing.

Theorem 13. *For any a, if $a > 0$ then $-a < 0$*

Another common shortcut, when two proofs are very similar, is that it's acceptable to simply skip a proof or a portion thereof, replacing it with an explanation that you've essentially already done the work and don't want to repeat it. Once you have proven Theorem 13, we can use this shortcut to prove Theorem 14 as follows.

Theorem 14. *For any a, if $a < 0$ then $-a > 0$*

Proof. Same as the proof of Theorem 13 except with $<$ and $>$ switched. □

Theorem 15. $a \leq b \Leftrightarrow (a < b$ *or* $a = b)$

Part D

Provide proofs for the following theorems.

Theorem 16. $a \geq b$ *or* $a < b$

Theorem 17. $a > b$ *or* $a \leq b$

Theorem 18. $1 > 0$

Part E

Provide proofs for the following theorems.

Theorem 19. *For any a, either $a < 0$, $a = 0$, or $a > 0$.*

Theorem 20. *For any a, b, if $a < 0$ and $b < 0$ then $ab > 0$.*

Theorem 21. *For any a, b, c, if $a < b \leq c$ or $a \leq b < c$ then $a < c$.*

Note: The notation $a < b < c$ and its friends are shortcut for conjunctions, such as "$a < b$ and $b < c$," but *Lurch* cannot handle this.

Part F

Provide proofs for the following theorems.

Theorem 22. *For any a, b, if $a \geq 0$ and $b \geq 0$ then $a + b \geq 0$.*

Theorem 23. *For any a, b, if $a < 0$ and $b < 0$ then $a + b < 0$.*

Theorem 24. *For any a, b, $(-a) + (-b) = -(a + b)$. That is, the additive inverse of $a + b$ is the sum of the additive inverses of a and b*

Hint: A chain of equalities will probably be helpful in proving Theorem 24.

Chapter 11

Mathematical Induction

11.1 Why introduce induction?

One of the most famous proof techniques in all of mathematics is called "Mathematical Induction." This chapter learns this new proof technique, because it will be necessary for several of the facts we wish to prove in the next chapter, in the realm of calculus.

As you may recall from Chapter 2, an inductive argument is one that extrapolates a conclusion from a set of evidence. In that chapter, we saw an example where several years of rain in San Diego in January led us to conclude that it would always rain in San Diego in January. An inductive argument is not an iron-clad argument, because it is merely extrapolating a pattern, rather than giving any reasons why the pattern must surely continue. It could be the case that rare conditions might eventually occur that prevent rain for an entire January in San Diego. Or it could be that global climate will eventually change drastically, so that rain in San Diego will be the exception rather than the norm. Thus inductive arguments are not airtight; they do not nearly approach the certainty of the deductive arguments you've done in all your proofs in this textbook.

So why would we introduce induction into mathematics? The answer is that *mathematical* induction is not really an inductive argument at all; rather, it just has some things in common with inductive arguments, and thus received that name. But it is another type of deductive argument, just one that has hints of induction in its style.

11.2 One new rule

Mathematical induction is a single new deductive rule that you will be permitted to use in your proofs. The version introduced below can prove statements only about *all natural numbers.* The natural numbers, usually written $\mathbb{N}$, is the collection of nonnegative whole numbers: 0, 1, 2, 3, and so on forever. Allow me to introduce the rule in two forms, and then discuss its properties. You will notice that it is a more complex rule than any we have seen so far, and thus it is not easy to understand it upon first reading it; the explanations below the rule can help clarify its purpose and use.

In the formal style in which we learned all the rules for SL and QL, mathematical induction can be expressed as follows. In stating this rule, I use a new notation, $k \in \mathbb{N}$, which means "k is in the set of natural numbers," or simply "k is a natural number." We usually pronounce it just as "k is in $\mathbb{N}$," and we will study the $\in$ symbol more in Chapter 13.

m. $\mathcal{A}[n = 0]$

n. $\forall k, (k \in \mathbb{N} \wedge \mathcal{A}[n = k] \Rightarrow \mathcal{A}[n = k + 1])$

$\forall n, (n \in \mathbb{N} \Rightarrow \mathcal{A})$ Mathematical Induction *m.*, *n.*

In the less formal style we've adopted in recent chapters, we might summarize the rule as follows. To shorten the notation, I use the common shorthand $\forall k \in \mathbb{N}, \mathcal{A}$ as shorthand for $\forall k, (k \in \mathbb{N} \Rightarrow \mathcal{A})$.

Definition 25 (mathematical induction)**.** If $\mathcal{A}[n = 0]$ and $\forall k \in \mathbb{N}$, $\mathcal{A}[n = k]$ implies $\mathcal{A}[n = k + 1]$, then $\forall n \in \mathbb{N},\ \mathcal{A}$.

Before we discuss the rule, allow me to point out one particular notational shortcut that is used in it. In this rule, I wrote "$\forall k \in \mathbb{N}$," which is not a construct we'd seen before. In English, we might think of this as saying, "for every natural number k," but in logical language, we can see its meaning by looking back at the formal version of the rule, above. Specifically, we translate $\forall k \in \mathbb{N}, \mathcal{A}$ into $\forall k, (k \in \mathbb{N} \Rightarrow \mathcal{A})$. I will use this shorthand freely from this point on in the textbook, and you are permitted to use it as well.

The first thing to notice about the rule itself is that it allows you to conclude statements of the form $\forall n \in \mathbb{N},\ \mathcal{A}$. That is, you should only use mathematical induction if you're trying to prove that a certain statement is true for all natural numbers. For example, if you were attempting to prove the statement $\forall n \in \mathbb{N},\ n^2 \geq 0$, you might proceed by mathematical induction. The first example (later in this chapter) of using the rule will prove the statement

$$\forall n \in \mathbb{N},\ \sum_{i=0}^{n} i = \frac{n(n+1)}{2}.$$

Second, let's understand the two premises that we are supposed to prove in order to obtain the conclusion that the statement $\mathcal{A}$ is true about all natural numbers n. The first premise, $\mathcal{A}[n = 0]$, means that the statement $\mathcal{A}$ must be true when we substitute 0 in for n. This is almost always very easy to prove, because we take some complicated mathematical formula and plug in 0 for the variable, resulting in a formula in which much of the algebra usually simplifies or outright vanishes. You will see this in examples later in this chapter, but I'm sure that you can already imagine that replacing a variable with zero simplifies things.

The second premise is more complicated. To establish such a premise, we must prove that $\forall k \in \mathbb{N}$, if the statement $\mathcal{A}$ is true about k, then it is true about $k + 1$. This statement is where the word "induction" comes from. It says that knowing that the fact $\mathcal{A}$ is true about some number k implies that it must continue to be true also about the next natural number, $k + 1$. Let's consider a very simple example to build our intuition for this.

Imagine that $\mathcal{A}$ is the phrase "the natural number n has been painted blue." Of course, one cannot paint abstract objects, but let's imagine for a moment that we could get our hands on the natural numbers and paint them blue. Establishing the first premise, $\mathcal{A}[n = 0]$, would mean that the natural number 0 has been painted blue. Please update your mental picture of the natural numbers so that 0 is blue, perhaps even freshly dripping paint. Now imagine further that the second premise is true: For every natural number k that has been painted blue, the next number $k+1$ has also been painted blue. Since 0 is blue, you are forced to now imagine that 1 is blue. And since you now see 1 as blue, you are forced to see 2 as blue. And then 3, and so on, forever.

This is the idea of mathematical induction. We prove a statement is true about zero, then we prove that the truth of the statement propagates upwards through the entire chain of natural numbers. Mathematical induction allows us to conclude from this that the statement does indeed hit the *entire* chain of natural numbers, so we can conclude that the statement is true about every $n \in \mathbb{N}$. We'll see an example next.

Quiz Yourself

- Where did we first see the notation $\mathcal{A}[n = k]$, and what does it mean?
- Verify that you understand the equation above by computing both $\sum_{i=0}^{3} i$ and $\frac{3(3+1)}{2}$ and getting the same result for both.

11.3 A first induction theorem

We wish to prove the following theorem, as promised in the previous section.

Theorem 26. *For any* $n \in N$, $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

We will do so by mathematical induction. When using induction, there is a commonly accepted way to structure the proof, to help the reader know that you're using induction, and to see the structure of your argument. I write the following proof *showing only that structure,* and leaving all other portions of the proof incomplete. Thus the following "proof" is really just a proof outline that we will discuss first, and then fill in thereafter.

Proof. This proof will be done by mathematical induction.

Base case: We must prove that

$$\sum_{i=0}^{0} i = \frac{0(0+1)}{2}.$$

(We must place here a proof of that fact.)

Induction step: Assume the theorem is true at k, in other words,

$$\sum_{i=0}^{k} i = \frac{k(k+1)}{2}.$$

We must prove that the theorem is still true at $k+1$, in other words,

$$\sum_{i=0}^{k+1} i = \frac{(k+1)(k+2)}{2}.$$

(We must place here a proof of that fact.)

By mathematical induction, the theorem is therefore true for all natural numbers n. □

The above "proof," as you can tell, is quite incomplete. It's missing some of its most important parts! However, I have left them out so that you can see this proof as a template that can be used for all induction proofs. Here are the specific components of the above proof that you should repeat in your own proofs that rely on mathematical induction:

1. Begin the proof with a sentence that tells the reader you're going to use mathematical induction.
2. Have a paragraph labeled as the "base case," which means the first premise, in which you substitute 0 for n. Prove that premise first.
3. Have a paragraph labeled as the "induction step," which means the second premise, in which you must prove that $\mathcal{A}[n = k]$ implies $\mathcal{A}[n = k + 1]$. Recall that it is called the induction step for the reasons described above; it is what propagates evidence about smaller natural numbers up to larger ones.
4. In your induction step, be sure to clearly state the assumption $\mathcal{A}[n = k]$ and the goal $\mathcal{A}[n = k + 1]$. Recall that the assumption is available for you to use, while the goal is what you must prove. The induction step is really an informal use of the $\Rightarrow$I rule, since the second premise contains an if-then statement.
5. End the proof by stating that you have completed your mathematical induction argument, and thus your proof is done.

11.4 Completing the example with new privileges

In order to complete the above proof template (turning it into an actual proof) we will need to use a good bit of algebra. As you've seen from Chapter 10, proving facts of algebra is very hard! We therefore introduce three new privileges, which you have earned through all your hard work in Chapter 10.

1. You can use the reason "by arithmetic" to justify any step of arithmetic students learn in elementary school. For instance, to prove that $7 + (15 - 1) = 21$ you do not need to go through arduous steps from the axioms; simply say "by arithmetic" and be done.
2. You can use the reason "by algebra" to justify anything an eighth-grade student would accept as valid algebra (when working correctly).

   To justify something simple, such as $a(b + c) - ab = ac$, you can just cite "algebra," because that statement is simple, true, and a matter of algebra alone. To be sure you're doing it correctly, you might consider getting out some scrap paper and quickly scratching down a chain-of-equations proof that uses only the axioms, to be sure that you're following them. But as long as you are, there's no longer any reason to show them in your final work.

   But be careful! Do not simply write something like $a \cdot \frac{1}{a} = 1$ and cite algebra! If you were to check that against the axioms, you would see that it requires the assumption that $a \neq 0$, which you must be sure to have. So with this new privilege comes the responsibility to use it carefully.
3. You are also permitted to write algebraic expressions in more common, more compact forms. You may write $a - b$ as an abbreviation for $a + (-b)$, you may write $\frac{a}{b}$ as an abbreviation for $a \cdot \frac{1}{b}$, and you may write ab as an abbreviation for $a \cdot b$.

But there remains one tool we need if we are to complete the proof template from the previous section. We must also learn how to deal with the summation symbol.

Definition 27 (summation)**.** The following two facts together define the summation symbol and its uses in our proofs.

1. To sum up just one item does not require any summing at all.

$$\sum_{i=a}^{a} f(i) = f(a)$$

2. The final item in a summation can be separated out as its own term.

$$\sum_{i=a}^{b} f(i) = \left(\sum_{i=a}^{b-1} f(i)\right) + f(b)$$

The first part of this definition will be useful in the base case of the induction proof, and the second part will be useful in the induction step. Let's see the final proof now.

Proof. This proof will be done by mathematical induction.

Base case: Prove that

$$\sum_{i=0}^{0} i = \frac{0(0+1)}{2}.$$

By arithmetic, $\frac{0(0+1)}{2} = 0$. By the definition of Σ, $\sum_{i=0}^{0} i = 0$. By substitution, the theorem is true in this case.

Induction step: Assume the theorem is true at k, in other words,

$$\sum_{i=0}^{k} i = \frac{k(k+1)}{2}.$$

I have to prove that the theorem is still true at k+1, in other words,

$$\sum_{i=0}^{k+1} i = \frac{(k+1)(k+2)}{2}.$$

$$\begin{aligned}
\sum_{i=0}^{k+1} i &= \left(\sum_{i=0}^{k} i\right) + (k+1) && \text{by definition of } \Sigma \\
&= \frac{k(k+1)}{2} + k + 1 && \text{induction hypothesis} \\
&= \frac{k(k+1)}{2} + \frac{2(k+1)}{2} && \text{algebra} \\
&= \frac{(k+1)(k+2)}{2} && \text{algebra}
\end{aligned}$$

So the theorem is true in this case as well. Therefore it is true for all natural numbers n by mathematical induction. □

Although *Lurch* is capable of handling proofs by induction, it does not yet come with rules for summation included. Consequently, from this point forward in the textbook, I do not include instructions on how to do the work in *Lurch*. My hope is that by this point you will have graduated from the need for constant feedback from *Lurch* and have sufficient confidence in your own critical assessment of your own proofs that you no longer need *Lurch* to do it for you.

Quiz Yourself

- Where can you find a proof template to use in many of the exercises in this chapter?
- How many gaps are there in that template that you must fill in?
- What two new shortcut rules of deduction do you have for use in this chapter's exercises?

Practice Exercises

Part A

Prove the following fact using mathematical induction. A proof template has been created for you.

Theorem 28. *For all $n \in \mathbb{N}$, if $n \geq 4$ then $n! \geq 2^n$.*

Proof. By mathematical induction, starting with $n = 4$.

Base case: When $n = 4$, we must prove that $4! \geq 2^4$.

(You must place here a proof of that fact.)

Induction step: The induction hypothesis is that $k! \geq 2^k$, and so we must prove that $(k+1)! \geq 2^{k+1}$.

(You must place here a proof of that fact.)

So the theorem is true in this case as well. Therefore it is true for all natural numbers n by mathematical induction. □

Part B

Prove the following two theorems using mathematical induction.

Theorem 29. *The sum of the first n powers of two is one less than the next one. That is, for all $n \geq 1$,*

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1.$$

Theorem 30. *The sum of the first n odd numbers is n^2. That is, for all $n \geq 1$,*

$$\sum_{i=1}^{n} (2i - 1) = n^2.$$

Part C

Prove the following three theorems using mathematical induction. Take particular care with the algebra involving fractions, to ensure that you follow the axioms.

Theorem 31. *The sum of the first n perfect squares is $\frac{n(n+1)(2n+1)}{6}$. That is, for any $n \geq 1$,*

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}.$$

Theorem 32. *The first n terms in a geometric series sum to $\frac{r^{n+1}-1}{r-1}$. That is, for any $n \in \mathbb{N}$ and $r \in \mathbb{R}$, if $r \neq 1$ and $r \neq 0$ then*

$$\sum_{i=0}^{n} r^i = \frac{r^{n+1} - 1}{r - 1}.$$

Theorem 33. *For all $n \geq 1$,*

$$\sum_{i=1}^{n} \frac{1}{(2i-1)(2i+1)} = \frac{n}{2n+1}.$$

Part D

Use the following definition to prove the theorem that follows it.

Definition 34 (divisibility)**.** For any natural numbers a and b, we write $a|b$ to mean $\exists n,\ (n \in \mathbb{N} \wedge an = b)$.

Theorem 35. *For all $n \in \mathbb{N}$, $3|(n^3 + 2n)$.*

Chapter 12

Calculus

Since Chapter 10, we've been applying the logic learned in earlier chapters to do mathematics. We began with very low-level mathematics, building tiny theorems from axioms. In Chapter 11 we moved up to more complex theorems of mathematics. And in these final three chapters of the text, we will address theorems from college-level mathematics.

We will prove in this chapter that some of the foundations of calculus are true, and in subsequent chapters go beyond calculus. If you ever wondered, when taking a calculus course, where the rules you memorized came from, you will find that answer in this chapter.

12.1 Absolute Value

We begin by establishing a few basic facts about absolute value, which we did not need before now. As in the past, I will provide you a definition for the absolute value operation, an associated privilege to make it easier to use, and then some theorems to try to prove related to it.

Definition 36 (absolute value)**.** We write $|x|$ to mean the absolute value of x. It satisfies the equation $|x| = x$ if $x \geq 0$ and the equation $|x| = -x$ if $x < 0$.

You can prove theorems on absolute value in *Lurch*, but as of this writing, *Lurch* does not support the notation $|x|$, so you must write abs(x) instead. This is less convenient, but if you want to use *Lurch* to check your work on theorems related to absolute value, you can do so.

You may want to use the following new privilege as you begin proving facts about absolute value. You'll notice that Definition 36 has two halves, one for if $x \geq 0$ and one for if $x < 0$. You will often, then, proceed by leveraging Theorem 16 from page 149 to state that $x \geq 0$ or $x < 0$, and then proceeding to use an argument structured according to the $\vee*$ rule.

As you do so, it is not necessary to cite the $\vee*$ rule explicitly. Simply state in English that you'll be considering the two cases separately. Then do each case in a separate paragraph, beginning with "Case 1:" and "Case 2:" respectively, or some similar phrasing. This helps the reader see the structure of your argument without your needing to cite a rule of logic explicitly.

I prove one example proof about absolute value in the text here, and I leverage exactly this new privilege. Consider this example when you try the first problems in the Practice Exercises section for this chapter.

Theorem 37. $|a| < b \Rightarrow a < b$

Proof. We consider two cases, as given by Theorem 16: either $a \geq 0$ or $a < 0$.

In the first case, Definition 36 gives us that $|a| = a$, so we can go from $|a| < b$ to $a < b$ simply by using substitution.

In the second case, Definition 36 gives us that $|a| = -a$. Since $a < 0$ in this case, Theorem 14 tells us that $-a > 0$. By Theorem 21 we therefore have $a < -a$. Substitution gives us $a < |a|$, and thus since $|a| < b$ is a given, Theorem 21 once again gives us $a < b$.

In both cases, we therefore have $a < b$. □

Quiz Yourself

- What was the criterion we gave in Chapter 10 for how to determine, when reading a proof in an informal style, whether it is correct?
- Try applying that criterion to Theorem 37.
- Jump ahead and try some of the Practice Exercises in this chapter on absolute values. (They will be leveraged in the proofs of more complicated theorems later in this chapter.)

12.2 Working with Limits

Students who have had calculus should remember that it concerns itself primarily with building three things on top of students' previous experience with algebra and trigonometry; those three new topics are limits, derivatives, and integrals. A more subtle point is that limits are the key concept, and derivatives and integrals are defined in terms of limits, as we will see in this chapter. Thus to proceed to calculus, the key step is for us to define a limit.

I will continue to use the shorthand from the previous chapter, writing expressions such as $\forall\varepsilon > 0, \mathscr{A}$ as shorthand for $\forall\varepsilon, (\varepsilon > 0 \Rightarrow \mathscr{A})$. Furthermore, statements such as "Let $\varepsilon > 0$ be arbitrary" will be shorthand for the following two steps in succession: Let ε be arbitrary. Assume $\varepsilon > 0$.

Definition 38. We write $\lim\limits_{x \to a} f(x) = L$ if and only if

$$\forall\varepsilon > 0,\ \exists\delta > 0,\ \forall x \in \mathbb{R},\ (0 < |x - a| < \delta \Rightarrow |f(x) - L| < \varepsilon).$$

This definition is quite complicated! It helps to translate each part of it into ordinary English, to help us understand the large pile of quantifiers involved. Such a piece-by-piece translation appears below, and is illustrated in Figure 12.1. But first notice some essential facts about the variables involved.

Because a and δ (the Greek letter delta) are compared to x values, they should be interpreted as horizontal measurements, as in the graph of f in Figure 12.1. And because L and ε (the Greek letter epsilon) get compared to $f(x)$ values, they should be interpreted as vertical measurements, as shown in the same figure. Thus we can summarize the meaning of Definition 38 as follows.

$\forall\varepsilon > 0,$	No matter how small a vertical window you choose,
$\exists\delta > 0,$	I can find a small enough horizontal window so that
$\forall x \in \mathbb{R},$	for every point on the x axis,
$(0 < \|x - a\| < \delta \Rightarrow$	if x is within δ of a (but not $x = a$),
$\|f(x) - L\| < \varepsilon)$	then $f(x)$ is within ε of the limit.

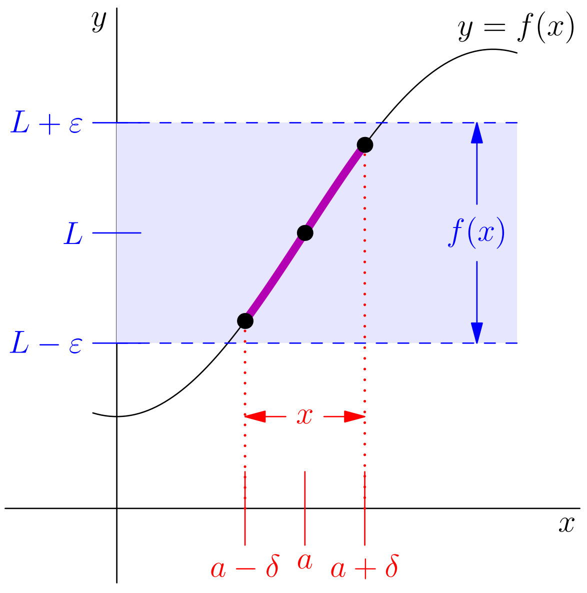


Figure 12.1: Illustration of the concepts in Definition 38.

In other words, if you pick a tiny window around the limit on the y-axis, I can find a tiny window around a on the x-axis so that all the $f(x)$ in that range lie within your chosen window. This is what Figure 12.1 shows.

We can use the above definition to prove several formulas about limits. We have therefore reached an exciting part of our text, because we are using our expertise in logic to prove the reliability of college-level mathematics! I prove two example theorems here, and leave further proofs about limits to the Practice Exercises section of this chapter.

Before we see two proofs using Definition 38, let's consider the general structure that such proofs will take. We're trying to prove the complex statement from that definition, which has a very specific structure of quantifiers and assumptions. As you know from your work in logic from earlier in this text, that specific structure will govern how we form the proof of the statement. For instance, the statement begins with $\forall\varepsilon > 0$, and thus our proof must have the following overall structure.

> Let ε be arbitrary. Assume $\varepsilon > 0$.
> *(Here, do a proof that $\exists\delta > 0,\ \forall x \in \mathbb{R},\ (0 < |x - a| < \delta \Rightarrow |f(x) - L| < \varepsilon)$.)*
> Because ε was an arbitrary positive number, we conclude that $\forall\varepsilon > 0,\ \exists\delta > 0,\ \forall x \in \mathbb{R},\ (0 < |x - a| < \delta \Rightarrow |f(x) - L| < \varepsilon)$.
> By Definition 38, that means that $\lim\limits_{x\to a} f(x) = L$.

To see how all of the above came about, notice that there are silent uses of the $\forall$I and $\Rightarrow$I rules.

When we look at the italicized statement in the proof template above, we see that the portion of the proof that remains undone is a statement that begins with an $\exists$, followed by a $\forall$, followed by an $\Rightarrow$. We therefore flesh out the proof further by applying the corresponding three introduction rules, $\exists$I, $\forall$I, and $\Rightarrow$I. The result is as follows. (I explain the blanks below.)

Let ε be arbitrary. Assume $\varepsilon > 0$.
Let x be arbitrary. Assume $x \in \mathbb{R}$ and $0 < |x - a| <$ ______.
(Here, do a proof that $|f(x) - L| < \varepsilon$.)
From our assumption that $0 < |x - a| <$ ______, we find that we have proven $0 < |x - a| <$ ______ $\Rightarrow |f(x) - L| < \varepsilon$.
Therefore $\exists \delta > 0,\ (0 < |x - a| < \delta \Rightarrow |f(x) - L| < \varepsilon)$.
Because ε was an arbitrary positive number, we conclude that $\forall \varepsilon > 0,\ \exists \delta > 0,\ \forall x \in \mathbb{R},\ (0 < |x - a| < \delta \Rightarrow |f(x) - L| < \varepsilon)$.
By Definition 38, that means that $\lim\limits_{x \to a} f(x) = L$.

Recall that the $\exists$I rule requires that the statement in question be proven about some actual value first, before the rule can be applied. For example, if we wish to prove $\exists x,\ x > 0$, we should first prove a statement like $1 > 0$ (as you did in Theorem 18), and then apply $\exists$I to conclude $\exists x,\ x > 0$.

Similarly, in the proof outline above, you're expected to fill in the blank with some actual constant, variable, or algebraic expression—whatever constant will enable you to most easily complete your proof. Let's see an example.

Theorem 39. *For any c, $\lim\limits_{x \to c} x = c$.*

To use the outline from above in this situation, we must replace $f(x)$, a, and L with the appropriate values for this proof. In this case, we have $f(x)$ is simply x, the limit L is c, and a is also c. This gives us the following proof outline.

Proof. Let ε be arbitrary. Assume $\varepsilon > 0$.

Let x be arbitrary. Assume $x \in \mathbb{R}$ and $0 < |x - c| <$ ______.

(Here, do a proof that $|x - c| < \varepsilon$.)

From our assumption that $0 < |x-c| <$ ______, we find that we have proven $0 < |x-c| <$ ______ $\Rightarrow |x-c| < \varepsilon$. Therefore $\exists \delta > 0,\ (0 < |x - c| < \delta \Rightarrow |x - c| < \varepsilon)$. Because ε was an arbitrary positive number, we conclude that

$$\forall \varepsilon > 0,\ \exists \delta > 0,\ \forall x \in \mathbb{R},\ (0 < |x - c| < \delta \Rightarrow |x - c| < \varepsilon).$$

By Definition 38, that means that $\lim\limits_{x \to c} x = c$. ☐

In order to complete the above proof outline, we need to choose a mathematical expression to place in the blank that will make it easy to complete the proof. In this first example, the answer is quite easy: Our goal is to prove that $|x - c| < \varepsilon$, so we should place ε in the blank, giving us our goal as a free assumption! Obviously not all proofs of limit equations will be as easy as this first example, but here is the completed version.

Proof. Let ε be arbitrary. Assume $\varepsilon > 0$.

Let x be arbitrary. Assume $x \in \mathbb{R}$ and $0 < |x - c| < \varepsilon$.

Since we assumed that $0 < |x - c| < \varepsilon$, we know that the simpler statement $|x - c| < \varepsilon$ is true, and so we have proven $0 < |x - c| < \varepsilon \Rightarrow |x - c| < \varepsilon$. Therefore $\exists \delta > 0,\ (0 < |x - c| < \delta \Rightarrow |x - c| < \varepsilon)$. Because ε was an arbitrary positive number, we conclude that

$$\forall \varepsilon > 0,\ \exists \delta > 0,\ \forall x \in \mathbb{R},\ (0 < |x - c| < \delta \Rightarrow |x - c| < \varepsilon).$$

By Definition 38, that means that $\lim\limits_{x \to c} x = c$. ☐

Let's see one other, slightly more complicated example of proving a limit equation. In this second example, I do not show you an outline first and fill in the blank. Rather, I just write the theorem and proof out, and suggest that you study them carefully yourself to see how they fit the form taught earlier in this section.

Theorem 40. *For any constant c and function f, if $\lim_{x\to a} f(x) = L$ then $\lim_{x\to a} cf(x) = cL$.*

Proof. Let a and c be arbitrary and f be an arbitrary function. Assume $\lim_{x\to a} f(x) = L$. Then by Definition 38, we have

$$\forall\varepsilon > 0,\ \exists\delta > 0,\ \forall x \in \mathbb{R},\ (0 < |x - a| < \delta \Rightarrow |f(x) - L| < \varepsilon). \tag{12.1}$$

We want to prove that $\lim_{x\to a} cf(x) = cL$, which means proving that $\forall\varepsilon > 0,\ \exists\delta > 0,\ \forall x \in \mathbb{R},\ (0 < |x - a| < \delta \Rightarrow |cf(x) - cL| < \varepsilon)$. So take any $\varepsilon > 0$.

Now use (12.1) to get a $\delta > 0$ be such that $\forall x \in \mathbb{R},\ (0 < |x - a| < \delta \Rightarrow |f(x) - L| < \frac{\varepsilon}{|c|})$. Take any x and assume $0 < |x - a| < \delta$, so that we can conclude $|f(x) - L| < \frac{\varepsilon}{|c|}$.

Note that $0 \le |f(x) - L|$ because the right hand side is an absolute value (Theorem 48). And since $|f(x) - L| < \frac{\varepsilon}{|c|}$, we have $\frac{\varepsilon}{|c|} > 0$ by transitivity (Theorem 21).

$$\begin{aligned}
|cf(x) - cL| &= |c(f(x) - L)| && \text{algebra}\\
&= |c||f(x) - L| && \text{Theorem 50}\\
&< |c| \cdot \frac{\varepsilon}{|c|} && \text{order axioms}\\
&= \varepsilon && \text{algebra}
\end{aligned}$$

□

Quiz Yourself

- In the equation $\lim_{x\to 2} x^2 = 4$, what are the a, $f(x)$, and L from Definition 38?
- Play the following game with a partner: Let him or her choose $\varepsilon > 0$, then you choose $\delta > 0$, then he or she chooses $x \in \mathbb{R}$. Can you guarantee by your choice of δ that $0 < |x - a| < \delta \Rightarrow |f(x) - L| < \varepsilon$? Because $\lim_{x\to 2} x^2 = 4$ is true, you should be able to if you play correctly.
- In the equation $\lim_{x\to 1} \frac{x}{|x|} = 1$, what are the a, $f(x)$, and L from Definition 38?
- Play the same game as above. But in this case, because $\lim_{x\to 1} \frac{x}{|x|}$ is false, your partner should always win if he or she plays correctly.
- Explain how the various steps in the game relate to the elements of Figure 12.1.

You are now able to complete the Practice Exercises on limit equations at the end of this chapter.

12.3 The foundations of calculus

This is it! You have reached the point in the text where you are able to prove several of the foundational theorems of calculus. In fact, if you've done any of the exercises on limit equations, then you've already proven some of them. But the first and most common application of limits is to derivatives, which we now define.

Definition 41. If f is a function then its derivative f' is a function defined as follows. For any $x \in \mathbb{R}$, if

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h} = L,$$

then $f'(x) = L$.

If you have had a calculus course before, you might recognize that the definition here is no different than it is in most calculus textbooks. The reason for this is that we've spent the time learning about the algebra of sums, differences, fractions, and functions, and then building limits on top of them.

It may seem a little strange to write Definition 41 as you see it above. Why did I not just write

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}?$$

The reason is because not all limits exist, and you will only be able to prove

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h} = L$$

from Definition 38 if the limit actually exists. If not, then the left-hand side of the if-then statement in Definition 41 is not true, and so $f'(x)$ remains undefined for that value of x.

Let us now see how this definition enables us to build some of the basic rules for differentiation. The following proof leverages the theorems about limits at the end of this chapter. All the hard work was done in proving the theorems on limits, and thus this calculus proof is quite brief. You will find, when proving theorems on derivatives yourself, that this same fact is true; they are much easier to prove than the theorems on limits.

Theorem 42. *If $f(x) = x$ then $f'(x) = 1$.*

Proof.

$$\begin{aligned}
f'(x) &= \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} && \text{Definition 41} \\
&= \lim_{h \to 0} \frac{x + h - x}{h} && \text{formula for } f \text{ assumed above} \\
&= \lim_{h \to 0} \frac{h}{h} && \text{algebra} \\
&= \lim_{h \to 0} 1 && \text{“algebra”} \\
&= 1 && \text{Theorem 51}
\end{aligned}$$

□

Because proofs of derivative rules are much easier than the other exercises in this chapter, I will do no further examples. Try the exercises now and see how far you've come in your study of both mathematics and logic!

Practice Exercises

Part A

Prove each of the following theorems. Note that the final one can be a short proof that heavily references the first four.

Theorem 43. *If $a \geq 0$ and $b \geq 0$ then $|a+b| \leq |a| + |b|$.*

Theorem 44. *If $a \geq 0$, $b < 0$, and $a + b \geq 0$, then $|a+b| \leq |a| + |b|$.*

Theorem 45. *If $a \geq 0$, $b < 0$, and $a + b < 0$, then $|a+b| \leq |a| + |b|$.*

Theorem 46. *If $a < 0$ and $b < 0$ then $|a+b| \leq |a| + |b|$.*

Theorem 47 (triangle inequality)**.** $|a+b| \leq |a| + |b|$

Part B

Prove the following simple theorems about absolute value. Recall the definition and privilege introduced in Section 12.1.

Theorem 48. $|a| \geq 0$

Theorem 49. *If $|a-b| < c$ then $|a| < |b| + c$.*

Part C

Prove the following more difficult theorem about absolute values.

Theorem 50. $|ab| = |a| \cdot |b|$

Hint: Break the possibilities up into the following six cases, again by Theorem 19.

1. $a = 0$ (and the value of b can be anything)
2. $b = 0$ (and the value of a can be anything)
3. $a > 0$ and $b > 0$
4. $a < 0$ and $b > 0$
5. $a > 0$ and $b < 0$
6. $a < 0$ and $b < 0$

Part D

Prove the following two short theorems about limits.

Theorem 51. $\lim\limits_{x \to a} b = b$

Theorem 52. $\lim\limits_{x \to c} x^2 = c^2$

Part E

Prove the limit sum rule, stated below. An important portion of the proof has been completed for you, and you are expected to fill in what's missing.

Theorem 53 (limit sum rule)**.** *For any functions f, g and any c, if $\lim_{x\to c} f(x) = L$ and $\lim_{x\to c} g(x) = M$, then $\lim_{x\to c}(f(x) + g(x)) = L + M$.*

Proof. Let c be arbitrary and assume $\lim_{x\to c} f(x) = L$ and $\lim_{x\to c} g(x) = M$, which means

$$\forall\varepsilon > 0,\ \exists\delta > 0,\ \forall x,\ (0 < |x - c| < \delta) \Rightarrow (|f(x) - L| < \varepsilon)$$

and

$$\forall\varepsilon > 0,\ \exists\delta > 0,\ \forall x,\ (0 < |x - c| < \delta) \Rightarrow (|g(x) - M| < \varepsilon).$$

We must prove

$$\forall\varepsilon > 0,\ \exists\delta > 0,\ \forall x,\ (0 < |x - c| < \delta) \Rightarrow (|(f(x) + g(x)) - (L + M)| < \varepsilon).$$

So let $\varepsilon > 0$ be arbitrary. From the two statements centered above, we can conclude the following two.

$$\exists\delta > 0,\ \forall x,\ (0 < |x - c| < \delta) \Rightarrow \left(|f(x) - L| < \frac{\varepsilon}{2}\right)$$

$$\exists\delta > 0,\ \forall x,\ (0 < |x - c| < \delta) \Rightarrow \left(|g(x) - M| < \frac{\varepsilon}{2}\right)$$

So let $\delta_1 > 0$ be such that $\forall x,\ (0 < |x - c| < \delta_1) \Rightarrow \left(|f(x) - L| < \frac{\varepsilon}{2}\right)$ and let $\delta_2 > 0$ be such that $\forall x,\ (0 < |x - c| < \delta_2) \Rightarrow \left(|g(x) - M| < \frac{\varepsilon}{2}\right)$.

Let x be arbitrary and assume $0 < |x - c| <$ ___________.

Fill in this gap in the proof.

Thus we can conclude that

$$\exists\delta > 0,\ \forall x,\ 0 < |x - c| < \delta \Rightarrow |(f(x) + g(x)) - (L + M)| < \varepsilon.$$

Since $\varepsilon > 0$ was arbitrary, we have proven this for all $\varepsilon > 0$, as required. □

Theorem 54 (limit product rule)**.** *For any functions f, g and any c, if $\lim_{x\to c} f(x) = L$ and $\lim_{x\to c} g(x) = M$, then $\lim_{x\to c}(f(x) \cdot g(x)) = LM$.*

Hint: This exercise is much harder than the previous. I suggest following the same outline as in the previous proof, but applying the two assumptions to create δ_1 and δ_2 satisfying the following inequalities.

$$\forall x,\ (0 < |x - c| < \delta_1) \Rightarrow \left(|f(x) - L| < \frac{\varepsilon/2}{|M|}\right)$$

$$\forall x,\ (0 < |x - c| < \delta_2) \Rightarrow \left(|g(x) - M| < \frac{\varepsilon/2}{L + \frac{\varepsilon/2}{|M|}}\right)$$

Part F

Prove the following fact by doing an ordinary proof about limits within a proof by mathematical induction.

Theorem 55. *For any natural number n, $\lim_{h\to 0}(x + h)^n = x^n$.*

Hint: Use the fact that $(x+h)^{n+1} = (x+h)(x+h)^n$.

Part G

Prove the following short theorems about derivative rules.

Theorem 56. *For any constant c, if $f(x) = c$, then $f'(x) = 0$.*

Theorem 57. *For any constant c and function f, if $g(x) = cf(x)$ then $g'(x) = cf'(x)$.*

Theorem 58 (derivative sum rule)**.** *For any functions f, g, if $k(x) = f(x) + g(x)$ then $k'(x) = f'(x) + g'(x)$.*

Part H

Prove the power rule for differentiation, as stated below.

Theorem 59 (power rule)**.** *If $f(x) = x^n$ then $f'(x) = nx^{n-1}$.*

Hints: Proceed by induction. Recall that $(x+h)^{n+1} = (x+h)(x+h)^n$. You will probably need to do a good bit of algebra, and to leverage Theorems 40, 53, and 55, as well as Definition 41.

Theorem 60 (derivatives of polynomials)**.** *The formula for the derivatives of polynomials is as follows.*

$$\left(\sum_{i=0}^{n} a_i x^i\right)' = \sum_{i=0}^{n-1} (i+1) a_{i+1} x^i$$

Hints: Proceed by induction. You will probably need most or all of the theorems on derivatives from the chapter and these exercises, as well as Definition 27.

Chapter 13

Abstract Mathematics

The previous few chapters showed that you could provide evidence for mathematical facts you already learned in other courses, perhaps even in high school. This chapter and the next combine to show you that you can prove some mathematical facts that you've never heard before, and that may surprise you.

This chapter lays the foundations by explaining how we can gather any quantity of mathematical objects together, into a collection called a *set,* and translate among sets using *functions.* The following chapter will then use these tools to show you how ordinary counting can be extended to infinite collections of things, with surprising results.

13.1 Sets

What is a set?

A set is any collection of items. We do not give a formal mathematical definition of a set, except that we add a new relation symbol to our language for expressing membership in sets, and we'll give rules for how to use it.

Definition 61 ($\in$, element)**.** If A is a set containing x, then we say that x is an *element* of A, and we write it $x \in A$. This is pronounced "x is an element of A" or just "x is in A."

An element is either in a set or not in the set. No element can be in a set more than once, nor in the set to a greater or lesser degree; it's just a true-or-false status. So the only relation in our language for expressing membership in a set is the symbol $\in$, defined above.

Writing sets

It is convenient to be able to express small sets with a simple notation. If we write a list of items between curly brackets, we mean by that the set containing exactly those items, and nothing else. Here are some examples.

- ▷ $\{1, 2, 3\}$ is a set with three elements. For example, $1 \in \{1, 2, 3\}$.
- ▷ $\{\text{Shirley Temple}, \text{Saddam Hussein}, \text{Katy Perry}\}$ is also a set (of people).

- ▷ $\{1, 2, 3, 4, 1, 2, 3, 4\}$ is a set but it only has four elements. An element can't be in a set twice, so this set is the same as $\{1, 2, 3, 4\}$.
- ▷ $\{10, 20, 30\}$ is the same set as $\{30, 20, 10\}$ because order is irrelevant. It's customary to write elements in a natural order, such as $\{10, 20, 30\}$, but that's just a convention with no actual mathematical significance.

You can make sets out of any collection of objects, even though we will usually think of sets containing mathematical objects. For instance, you could consider the set of desks in your classroom; that's a valid set.

You can also express sets with formulas in them that compute values. The set $\{1, 2-1, 3+1, 6-1, 6-2\}$ is obviously a set of numbers, but until we do the arithmetic, it's not clear what's in the set. After doing the arithmetic, we find that it is the set $\{1, 1, 4, 5, 4\}$, which is the same as $\{1, 4, 5\}$. It was just "in disguise" at first, until we computed its values.

Because you can put any object inside a set, you can even place sets inside other sets. For instance, we can consider the set $\{1, 2, \{3, 4\}\}$. Although it may *appear* as if this set has four elements, it does not. The set contains the three elements 1, 2, and $\{3, 4\}$. The first two are numbers, and the third is a set.

This is analogous to getting packages in the mail, where the outermost box may contain inner boxes. For instance, if you order two textbooks and a portable speaker from an online retailer, they may all come in one box. When you open the box, you see the three items you ordered. Now, if you open the box containing the portable speaker, you may find batteries, cables, and other miscellaneous items, but your order really consisted of three things.

The relation $\in$ for sets is only about the first level of contents of a set, like when you just open the box that comes in the mail. You see the three items you ordered, and those are the three elements of the set of things you ordered. Although your third item (the portable speaker) contains items within it, which are therefore *indirectly* in the box that was shipped to you, when we speak of sets, we'll only be speaking of what's *directly* inside the set.

For example, we can write $1 \in \{1, 2, \{3, 4\}\}$ and $2 \in \{1, 2, \{3, 4\}\}$, but we cannot write $3 \in \{1, 2, \{3, 4\}\}$; that would be a false statement. But we *can* write $\{3, 4\} \in \{1, 2, \{3, 4\}\}$! The set $\{3, 4\}$ is one of the things that is directly inside the larger set $\{1, 2, \{3, 4\}\}$.

This gets trickier when we consider sets like $\{\{1, 2, 3, 4\}\}$. This set contains only one element, the set $\{1, 2, 3, 4\}$. (Following the analogy with boxes, it would be like receiving a package with nothing inside it but another box, and all your items were inside that inner box.)

Even trickier, we might ask how many elements are in the set $\{x, y, z\}$. In that case, because x, y, and z are variables, it depends a lot on their value. If we were in a situation in which we had assumed $x = 1$, $y = 2$, and $z = 3$, then $\{x, y, z\} = \{1, 2, 3\}$, and there are three elements in the set. But there are situations in which $\{x, y, z\}$ would be a set with only *one* element in it. Can you think of such a situation?

We can summarize the notation introduced in this section by writing down a few axioms that you can use in proofs about sets. I formalize them as a definition here.

Definition 62 ($\{\ldots\}$ notation)**.** The meaning of the $\{\ldots\}$ notation used above is captured by the following logical sentences.

1. $x \in \{a\} \Leftrightarrow x = a$
2. $x \in \{a, b\} \Leftrightarrow x = a$ or $x = b$
3. $x \in \{a, b, c\} \Leftrightarrow x = a$ or $x = b$ or $x = c$
4. ...and so on, for any finite set. For example, $x \in \{a_1, a_2, \ldots, a_n\} \Leftrightarrow x = a_1$ or $x = a_2$ or $\ldots$ or $x = a_n$

Special Sets

Definition 63 ($\emptyset$, the empty set)**.** If we write nothing in between curly brackets, as in $\{\}$, it indicates a set with no elements. It is also commonly written with the special symbol $\emptyset$, which is just shorthand for $\{\}$. We can capture its meaning with the logical sentence $\forall x,\ x \notin \emptyset$, or equivalently, $\forall x,\ x \notin \{\}$. These sentences are saying "nothing is in the empty set."

In case it is not clear from Definition 63, you can use the equation $\emptyset = \{\}$ in your proofs if you need to; each notation is just shorthand for the other.

It is a common mistake to write $\{\emptyset\}$ when what you really mean is $\emptyset$, or $\{\}$. The set $\{\emptyset\}$ actually has one element in it, the element $\emptyset$! Think of $\{\emptyset\}$ as $\{\{\}\}$, which is a box with one thing in it—an empty box. By contrast, $\emptyset$ has no elements in it; it is the same as $\{\}$, and is analogous to a box with nothing it it at all—not even another (empty) box.

Since we will be using sets mostly to look at mathematics, we will want to have names for some of the most famous sets in mathematics, which show up in mathematical proofs all the time. The following definition introduces them, although rather informally. It is possible to give technical definitions of all of them, but we will not need to do so in this text.

Definition 64 (famous mathematical sets, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{Q}$)**.**

- ▷ The natural numbers are $\mathbb{N} = \{0, 1, 2, 3, 4, \ldots\}$.
- ▷ The integers are $\mathbb{Z} = \{\ldots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}$.
- ▷ The real numbers are $\mathbb{R}$.
- ▷ The rational numbers (all fractions of integers) are $\mathbb{Q}$.

Before you continue reading this chapter, it might be a good time to try Part A in the Practice Exercises, to be sure you understand what you've read so far.

Quiz Yourself

- How many elements are in the set $\{1, 1, 1, 1, 1, 1\}$?
- Consider the famous sets introduced in Definition 64. Does any of them include any of the others? (For example, is every natural number a real number?)

Operations on Sets

There are many common operations you can do with sets. I will briefly introduce intersection and union here, but not illustrate them in the way you would see in a textbook on probability or combinatorics. We will concentrate on defining them logically and using them in proofs. Although visualizations of set operations are excellent for building intuition, our focus here is on handling them in deductions.

Definition 65 (intersection, $\cap$)**.** If A and B are sets, then $A \cap B$ is their *intersection*, the set of elements that A and B have in common. That is, $x \in A \cap B \Leftrightarrow x \in A$ and $x \in B$.

It is easy to remember that the symbol $\cap$ means intersection, because you can think of the set of things two sets have in common as their "overlap," and the symbol $\cap$ looks like an archway "over" something. Here are two examples of the intersection of sets.

$$\{1, 2, 3, 4, 5\} \cap \{1, 3, 5, 7, 9\} = \{1, 3, 5\}$$

$$\{1, 2, 3\} \cap \{4, 5, 6\} = \{\}$$

In mathematical definitions, the symbol $\Leftrightarrow$ is often expressed with the English phrase "if and only if," which is often abbreviated "iff." I will use this abbreviation in the rest of this chapter and the next, as needed.

Definition 66 (union, $\cup$)**.** If A and B are sets, then $A \cup B$ is their *union*, the set of elements obtained by putting together all of A's elements and all of B's elements into one set. That is, $x \in A \cup B$ iff $x \in A$ or $x \in B$.

It is easy to remember that the symbol $\cup$ means union, because it looks like the letter U. Here are two examples of the union of sets.

$$\{1, 2, 3\} \cup \{1, 3, 5, 7\} = \{1, 2, 3, 5, 7\}$$

$$\{1, 2, 3, 4, 5\} \cup \{1, 3, 5\} = \{1, 2, 3, 4, 5\}$$

Equality and Subsets

Several of the proofs we'll do with sets involve demonstrating that two sets are equal. We therefore need a rule that defines when two sets are equal. The following does so, and I discuss its significance after the rule is stated.

Definition 67 (= for sets)**.** We say two sets A and B are equal if they contain exactly the same elements. That is, $A = B$ iff $\forall x,\ (x \in A \Leftrightarrow x \in B)$.

Notice that this rule determines when two sets are equal based *only* on the items that are in each. We can see embodied here two of the important concepts mentioned earlier in the chapter.

First, I mentioned that membership in a set is a true-or-false condition, not something that can be more or less true, or true twice or three times. In Definition 67, we see that membership in A and B is connected with the $\Leftrightarrow$ symbol, indicating that both $x \in A$ and $x \in B$ must be statements, and thus have either true or false values.

Second, I mentioned that elements do not have a particular order in which they appear in a set. For example, the set $\{1, 2, 3\}$ is the same as the set $\{1, 3, 2\}$. This can be seen by the fact that Definition 67 does not mention order anywhere. It judges two sets to be equal iff they have the same elements; order is not part of the judgment.

Definition 68 ($\subseteq$, $\subset$)**.** We say A is a *subset* of B if every element of A is an element of B. That is, $A \subseteq B$ iff $\forall x,\ (x \in A \Rightarrow x \in B)$. It is a *proper subset* if it's not equal to B, and we write that $A \subset B$.

Let's take a moment to consider some example sentences using these new relations, and assess their truth or falsity.

Consider the statement $\{1, 2, 3, 4, 5, 6, 7\} \subseteq \mathbb{N}$. Is it true or false? (Feel free to stop reading here and consider your answer for a moment before reading on!)

We evaluate the truth of that statement by asking whether every element of $\{1, 2, 3, 4, 5, 6, 7\}$ is also an element of $\mathbb{N}$. Because all seven of those numbers are indeed natural numbers, the answer is yes, $\{1, 2, 3, 4, 5, 6, 7\}$ is a subset of $\mathbb{N}$.

Now, is $\mathbb{N} \subset \mathbb{Z}$ true or false? Again, we ask ourselves if every element of $\mathbb{N}$ is also an element of $\mathbb{Z}$. That is, we're asking if every natural number is also an integer. Again, the answer is true, because to obtain $\mathbb{Z}$, you could start with $\mathbb{N}$ and then add in the negative whole numbers. Thus $\mathbb{Z}$ contains $\mathbb{N}$ and more.

Next, is $\mathbb{Q} \subseteq \mathbb{N}$ true or false? In this case, not every element of $\mathbb{Q}$ is an element of $\mathbb{N}$. Certainly, some things in $\mathbb{Q}$, such as $\frac{4}{2}$, are in $\mathbb{N}$, because $\frac{4}{2} = 2$, which is a natural number. But there are other elements of $\mathbb{Q}$, such as $\frac{1}{3}$, which are not natural numbers, because they are not even integers.

A trickier question is whether $\mathbb{R} = \mathbb{Q}$. For a long time, it was not known whether every number on the standard number line was rational. Eventually, mathematicians established that there are infinitely many *irrational* numbers, such as $\sqrt{2}$ and π, that are real numbers, but cannot be expressed as fractions, and are thus not in $\mathbb{Q}$.

13.2 Proofs about Sets

In the Practice Exercises for this chapter, you'll be asked to prove several facts about sets. Therefore you should have an opportunity to see what some proofs in this topic look like before you have to attempt them on your own. I proof two theorems here, then leave others for you to try as exercises. A few of those exercises are similar to these theorems, so you may wish to refer back to them when constructing your own proofs.

When doing proofs in this chapter and in later chapters, feel free to use these two theorems as reasons in your proofs. And as before, once you have proven a theorem in the exercises, feel free to cite it in your future work.

Theorem 69. $A \cup B = B \cup A$

Proof. To show $A \cup B = B \cup A$ means (by the definition of $=$ for sets) that we must show $\forall x,\ (x \in A \cup B \Leftrightarrow x \in B \cup A)$. So let x be arbitrary, and I'll do each side of the biconditional separately.

Assume $x \in A \cup B$, and by the definition of $\cup$, this means that $x \in A$ or $x \in B$. This is obviously logically equivalent to $x \in B$ or $x \in A$, which gives us that $x \in B \cup A$ (again by the definition of $\cup$). So $x \in A \cup B$ implies $x \in B \cup A$.

The other direction of the biconditional is nearly identical to the first. $\square$

Theorem 70. $A \cap \emptyset = \emptyset$

Proof. To show $A \cap \emptyset = \emptyset$ means (by the definition of $=$ for sets) that we must show $\forall x,\ (x \in A \cap \emptyset \Leftrightarrow x \in \emptyset)$. So let x be arbitrary, and I'll do each side of the biconditional separately.

Assume $x \in A \cap \emptyset$, and by the definition of $\cap$, this means that $x \in A$ and $x \in \emptyset$. Obviously therefore I've shown that $x \in A \cap \emptyset$ implies $x \in \emptyset$.

Assume $x \in \emptyset$. Further assume, towards a contradiction, that $x \notin A \cap \emptyset$. The contradiction we're seeking comes from the definition of the empty set, which states that for any x, $x \notin \emptyset$. This contradicts our earlier assumption, and so we conclude $x \in A \cap \emptyset$. $\square$

Quiz Yourself

- Is $\mathbb{N}$ a subset of $\mathbb{Q}$? Is it a proper subset?
- What was the criterion we gave in Chapter 10 for how to determine, when reading a proof in an informal style, whether it is correct?
- Try applying that criterion to Theorem 69 or 70.

13.3 Functions

In Chapter 12, we used functions when computing limits and derivatives. In all of those cases, we assumed that functions took real number inputs and gave real number outputs. After all, we were using axioms that assumed everything we were working with was a real number. We will soon be expanding to other kinds of inputs and outputs, but first let's get precise about function notation in general.

Definition 71 (formulas for functions)**.** When we define a function using an equation, such as $f(x) = 3x^2$, we are essentially writing a $\forall$ statement. That example formula means $\forall x,\ f(x) = 3x^2$. Whenever we write $f(x) =$ any formula, you should read it as a universally quantified statement defining the function's behavior.

Definition 72 (piecewise definition)**.** Some functions are defined "piecewise," meaning that they have different formulas defining their behavior on different kinds of inputs. To express this, we use notation like that in the following example.

$$f(x) = \begin{cases} \sin x & \text{if } x \geq 0 \\ -x & \text{otherwise} \end{cases}$$

Writing such a definition is still a $\forall$ statement, but one with conditional statements inside it. The above example can be translated into the following logical expression.

$$\forall x,\ \Big((x \geq 0 \Rightarrow f(x) = \sin x) \wedge (\neg(x \geq 0) \Rightarrow f(x) = -x)\Big)$$

Whenever you see a piecewise-defined function, you can think of it in terms of conditionals, as in this example.

Now that we have other sets to work with, such as $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and all manner of smaller sets like $\{1, 2, 3\}$, we can discuss functions that take inputs from a set other than the real numbers (for example, perhaps just the integers) and give outputs other than real numbers (for example, perhaps just the rational numbers). We therefore need to expand our understanding of functions, as well as our notation for how to describe them.

If f is any function from a standard calculus course, as in Chapter 12, we can make the following statement about it.

$$f : \mathbb{R} \to \mathbb{R}$$

This means "f takes real number inputs, and gives real number outputs." It is read as "f sends $\mathbb{R}$ to $\mathbb{R}$" or "f maps $\mathbb{R}$ to $\mathbb{R}$."

We can also have functions that map between sets other than $\mathbb{R}$. Consider the following examples.

- ▷ Let $f : \mathbb{N} \to \mathbb{N}$ be defined by $f(n) =$ the smallest prime number dividing evenly into n.
- ▷ Let $g : \mathbb{N} \to \mathbb{Q}$ be defined by $g(n) = \frac{n}{n+1}$.
- ▷ Let $h : \{1, 2, 3, 4\} \to \{\text{even}, \text{odd}\}$ be as follows.

$$h(1) = h(3) = \text{odd} \qquad \text{and} \qquad h(2) = h(4) = \text{even}$$

The function f defined above takes natural number inputs and gives natural number outputs. It would not make sense to try to compute $f(3.5)$ or $f(\pi)$, because those numbers do not have prime factors. The function g defined above could take most real numbers as inputs, and its formula would still make sense. But in defining it, I specifically restricted it to taking only natural number inputs, because then I can guarantee that it gives only rational number outputs. The function h defined above is interesting because it has only finitely many inputs and only finitely many outputs. And its outputs are not even numbers; they're words!

As we begin talking about inputs and outputs of a function, the following terms will be useful. Most of them you have seen before, if you have had a calculus course.

Definition 73 (domain, codomain, range)**.** When $f : A \to B$, the set A is called the *domain,* the set B is called the *codomain,* and the subset of B that f can give as outputs is called the *range.* To be precise, the range is $\{f(a) \mid a \in A\}$, which is the same thing as $\{b \in B \mid \exists a \in A,\ f(a) = b\}$.

Let's see how these terms apply to some of the functions defined above. Because $f : \mathbb{N} \to \mathbb{N}$, the domain and codomain of f are both $\mathbb{N}$. But because f can only output prime numbers (and it can output all prime numbers), its range is the set of prime numbers. Because $h : \{1, 2, 3, 4\} \to \{\text{even}, \text{odd}\}$, the domain is $\{1, 2, 3, 4\}$ and the codomain is $\{\text{even}, \text{odd}\}$. Because both of the elements of the codomain are given as actual outputs from h, the codomain of h is the same as its range. This was not the case for f; its range was a subset of its codomain, but not the entire codomain.

In fact, it will always be the case that the range is a subset of the codomain, and thus the range will always be smaller than or equal to the size of the codomain. In fact, it will also always be the case that the range will be smaller than or equal to the size of the *domain.* Can you see why? If not, the following section may clarify it. If it's still not clear, don't worry; that's the topic of Chapter 14.

Illustrating Functions

When the domain and codomain are small finite sets, we can define functions with simple pictures. Let $T : \{a, b, c, d, e\} \to \{10, 20, 30\}$ be defined as shown below.

$$\begin{tikzcd}
a \arrow[r] & 10 \\
b \arrow[dr] & 20 \\
c \arrow[r] & 30 \\
d \arrow[uuur] & \\
e \arrow[uur] &
\end{tikzcd}$$

That picture is a visual way to represent that $T(a) = T(d) = 10$ and $T(b) = T(c) = T(e) = 30$.

Can you also express the function T using a piecewise definition? You should be able to with just one condition, and one "otherwise" case. Try to do the same for the function h.

Quiz Yourself

- Write the definition of $|x|$ (Definition 36) using the notation of a piecewise function, from Definition 72.
- What is the relationship between the codomain and the range? Is one always a subset of the other?

Practice Exercises

Part A

1. How many elements are in the set $\emptyset$?
2. How many elements are in the set $\{\emptyset\}$?
3. How many elements are in the set $\{\{\{\}\}\}$?
4. If $x \in \{1, 2, 3\}$ and $x \neq 1$, what can we conclude?
5. If we assume some statement P and from that assumption prove $x \in \emptyset$, what can we conclude?

Part B

Let's say I'm doing a proof in which there is a set called A. If I let x be arbitrary, then I assume $x \in \mathbb{N}$, and from that assumption I prove that $x \in A$, what can I conclude?

Part C

Let A stand for the set of whole numbers from 1 through 5.

1. Compute $A \cap \{1, 2, 3\}$ and $A \cap \{6, 7, 8\}$.
2. Compute $A \cup \{1, 2, 3\}$ and $A \cup \{6, 7, 8\}$.
3. Compute $A \cap \emptyset$ and $A \cup \emptyset$.

Part D

Prove the following two theorems about sets.

Theorem 74. $A \cap B = B \cap A$

Theorem 75. $A \cup \emptyset = A$

Part E

What are the domains, codomains, and ranges of the two functions g and T defined in Section 13.3?

Name a function from $\mathbb{R}$ to $\mathbb{R}$ whose range is the positive reals.

Part F

Prove the following three theorems about functions.

Theorem 76. *Let $f : \mathbb{N} \to \mathbb{N}$ be such that $f(n)$ is the first digit of n.*

Then $\exists n \in \mathbb{N},\ \exists m \in \mathbb{N},\ (f(n) = f(m) \wedge n \neq m)$.

Theorem 77. *Let $g : \mathbb{N} \to \mathbb{N}$ be given by $g(n) = 10n + 1000$.*

Then $\forall n \in \mathbb{N},\ \forall m \in \mathbb{N},\ (g(n) = g(m) \Rightarrow n = m)$.

Theorem 78. *Let $h : \mathbb{N} \to \mathbb{N}$ be as follows.*

$$h(n) = \begin{cases} \frac{n}{10} & \textit{if } n \textit{ is a multiple of 10} \\ 42 & \textit{otherwise} \end{cases}$$

Then $\forall n \in \mathbb{N},\ \exists m \in \mathbb{N},\ h(m) = n$.

Chapter 14

Equinumerosity

The title of this chapter is probably an unfamiliar word to most readers. Two sets are "equinumerous" if they have the same number of elements. So for example $\{1, 2, 3\}$ and $\{\text{Ed}, \text{Mary}, \text{René}\}$ are two equinumerous sets. This is a very simple idea that doesn't require advanced mathematics to study it, but we will be generalizing this simple idea to infinite sets as well, finding some surprising results. To do so, we need to look a little more closely at functions.

14.1 Types of Functions

Types of functions

Definition 79 (injective, or "into," or "1-to-1"). A function $f : A \to B$ is *injective* (or an injection) if each of the elements of A maps to a different element of B. In symbols,

$$\forall x \in A,\ \forall y \in A,\ (f(x) = f(y) \Rightarrow x = y).$$

Such functions are also called "1-to-1" (or one-to-one) functions. Another way to say it is that f maps A "into" B.

If you look back at the function T illustrated on page 172, you'll see that T is *not* injective, for the following reason. We can find two elements x and y in the domain of T such that $T(x) = T(y)$ and yet $x \neq y$. For instance, $T(a) = T(d)$, but $a \neq d$. If T had been injective, then $T(a) = T(d)$ would have guaranteed that $a = d$. In a picture of a function, like the one on page 172 for T, if a function is to be injective, it cannot have two arrows that point to the same element of the codomain. An injective function is illustrated below.

$$\begin{tikzcd}
a \arrow[r] & 10 \\
b \arrow[dr] & 20 \\
c \arrow[ur] & 30 \\
d \arrow[dr] & 40 \\
e \arrow[dr] & 50 \\
 & 60
\end{tikzcd}$$

Therefore a simple way to think of injective functions is that each element of the domain goes to a *unique* element of the codomain. This is why injective functions are sometimes called "one-to-one" functions. If a function is two-to-one, or three-to-one, or more at even a few elements of the domain, then it is not injective.

Notice that in order for a function to be injective, the codomain has to be at least as large as the domain. This is because each element of the domain must point to a *different* element of the codomain, so there is something like a same-size copy of the domain in the codomain. In the function illustrated above, the five-element domain $\{a, b, c, d, e\}$ corresponds to the five-element range $\{10, 20, 30, 50, 60\}$, and thus the codomain needs to have at least five elements. This is the first connection between functions and counting. *An injection tells us that the codomain is at least as big as the range.*

Definition 80 (surjective, or "onto")**.** A function $f : A \to B$ is *surjective* (or a surjection) if each of the elements of B has some element of A mapped to it. In symbols,

$$\forall y \in B,\ \exists x \in A,\ f(x) = y.$$

Looking back at the same picture of T from page 172, we can see that T is also not surjective. There exists an element of the codomain that is not pointed to from any element of the domain. There is no input x to T such that $T(x) = 20$. Another way to say a function is surjective is to say that its codomain equals its range. The illustration of an injective function above is also not surjective, because 40 is not in its range. Here follows an illustration of a surjective function.

$$\begin{tikzcd}
a \arrow[r] & 10 \\
b \arrow[dr] & 20 \\
c \arrow[uur] & 30 \\
d \arrow[uur] & \\
e \arrow[uur] &
\end{tikzcd}$$

Connecting this to counting, we can see that for a function to be surjective, the domain has to be at least as big as the codomain. The reason for this is that every element in the codomain needs at least one element of the range to map to it. Thus the three-element codomain $\{10, 20, 30\}$ in the illustration above needs a domain of at least three elements to point to it. In fact, even if the domain had only been $\{c, d, e\}$, we see that just that portion of the function would have been surjective onto $\{10, 20, 30\}$, but in this case we also have some other elements in the domain as well. This is our second connection between functions and counting. *A surjection tells us that the domain is at least as big as the codomain.*

We can also speak of injective and surjective functions from a typical algebra or calculus course. For instance, $f(x) = x^2$ is not injective, because it sends two different inputs (say, $+2$ and -2) to the same output (4). It is also not surjective, because there is no input that makes $f(x) = -1$. On the other hand, $f(x) = e^x$ is injective, because if $f(x) = f(y)$ then $e^x = e^y$, and we can simply apply ln to both sides to conclude that $x = y$. That gives us a very short proof, using algebra, that $f(x) = f(y) \Rightarrow x = y$, which is the definition of injective. However, still $f(x) = e^x$ is not surjective, because it, too, cannot output -1. A simple example of a function from algebra and calculus that is both injective and surjective is a line, such as $f(x) = 3x + 1$. You will have an opportunity to prove this in the Practice Exercises (Theorem 100).

Functions that are both injective and surjective have a special term that collects both properties into one.

Definition 81 (bijective)**.** A function is *bijective* (or a bijection) if it is both injective and surjective.

The terminology we've just learned enables us to rephrase Theorem 76 from the previous chapter more succinctly. Rather than saying $\exists n \in \mathbb{N},\ \exists m \in \mathbb{N},\ (f(n) = f(m) \wedge n \neq m)$, we can just say that f is not injective. Bijective functions play a central role in the study of equinumerosity, because they combine both of the two counting facts from earlier. We'll come back to that after one final function-related definition.

Composing functions

Definition 82 (composition)**.** If $f : A \to B$ and $g : B \to C$ then the *composition* of the two functions, $f \circ g$, is a function satisfying the following two conditions.

$$f \circ g : A \to C$$

$$\forall x \in A,\ f \circ g(x) = g(f(x))$$

You have probably encountered function composition in an algebra or calculus course in the past. For instance, if $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ according to the formulas $f(x) = 2x$ and $g(x) = \sin x$, then we can compose those two functions in either order. Composing $f \circ g$ would yield the following function.

$$f \circ g(x) = g(f(x)) = \sin 2x$$

But composing $g \circ f$ yields a different function.

$$g \circ f(x) = f(g(x)) = 2 \sin x$$

Notice that in order for function composition to make sense, the codomain of the first function in the composition must equal the domain of the second. If we had $f : \mathbb{N} \to \mathbb{Q}$ and $g : \mathbb{Q} \to \mathbb{R}$, we could only compose them as $f \circ g$, not as $g \circ f$. The function $f \circ g$ first does the f operation, yielding a result in $\mathbb{Q}$, which can then be plugged into g, which takes inputs in $\mathbb{Q}$; that makes sense.

But if we tried to compose in the other order, as in $g \circ f$, we would first be applying g, yielding a result in $\mathbb{R}$, which we could not necessarily then plug into f, which requires inputs only in $\mathbb{N}$. The reason we could compose $2x$ and $\sin x$ in either order was because both had domain and codomain equal to $\mathbb{R}$. That was a special case.

It is a common mistake to think that the formula for $f \circ g(x)$ is $f(g(x))$, but that is *incorrect.* Definition 82 writes $f \circ g(x) = g(f(x))$, reversing the order of f and g across the equation. The function $f \circ g$ is supposed to apply f first, then g, so we must write f in the innermost parentheses, so that we're requiring it to be applied first.

Since we will naturally be doing proofs later in this chapter using the notions of injectivity, surjectivity, and function composition, let's see an example proof that ties together a few of these notions.

Theorem 83. *The composition of two injections is an injection.*

Proof. Let $f : A \to B$ and $g : B \to C$ be two injections. We must prove that $f \circ g : A \to C$ is an injection. Definition 79 therefore requires us to prove that

$$\forall x \in A,\ \forall y \in A,\ (f \circ g(x) = f \circ g(y) \Rightarrow x = y).$$

Using Definition 82 and substituting into the equation immediately above, we find that our goal becomes

$$\forall x \in A,\ \forall y \in A,\ (g(f(x)) = g(f(y)) \Rightarrow x = y).$$

So let x and y be arbitrary elements of A and assume $g(f(x)) = g(f(y))$. Then because g is injective, we can conclude $f(x) = f(y)$, using Definition 79. And then because f is injective, we can conclude $x = y$ the same way. Thus for any x and y, $f \circ g(x) = f \circ g(y) \Rightarrow x = y$, and we've achieved our goal. $\square$

Quiz Yourself

- What does an injective function tell you about the relationship between the domain and the codomain?
- What does a surjective function tell you about the relationship between the domain and the codomain?
- If a function does not pass the "horizontal line test" then it is not injective. This test checks whether any horizontal line touches the function's graph more than once. Why is this a sensible test for injectivity?

14.2 Counting

At the beginning of this chapter, I pointed out that comparing the sizes of two small sets is quite easy. But we will be comparing the sizes not only of large sets, but of infinite ones. Thus we will need a more sophisticated tool than the simple counting one does with a few objects on a table.

As you saw in the previous section, we can use functions to compare the sizes of two sets. An injection $f : A \to B$ tells us that B is at least as big as A, possibly bigger. And a surjection $f : A \to B$ tells us the reverse; A is at least as big as B, possibly bigger. Therefore a bijection $f : A \to B$ tells us that two functions are exactly the same size. We formalize this idea with the following definition.

Definition 84 (equinumerous)**.** We say that two sets A and B have the same size (are *equinumerous*) if there is a bijection from A to B. We will write this as $A \sim B$.

Counting with sets and functions

As we learn to use sets and functions for counting, we will begin with the simplest case, counting finite sets of things. The following set is useful for comparing to other finite sets.

Definition 85 (I_n)**.** The set of the first n natural numbers, starting at 0, is called I_n.

$$I_n = \{0, 1, 2, \ldots, n-1\}$$

For example, $I_3 = \{0, 1, 2\}$, and $I_{50} = \{0, 1, 2, 3, 4, \ldots, 49\}$. $I_0 = \emptyset$.

Let's begin with a very simple example use of counting with finite sets. To which set I_n is the set $\{\text{Moe}, \text{Larry}, \text{Curly}\}$ equinumerous? Clearly, $\{\text{Moe}, \text{Larry}, \text{Curly}\}$ has three elements, so we would hope it would be equinumerous with I_3, which is $\{0, 1, 2\}$. Can we create a bijection between those two sets? Yes, we can do so easily, as shown here.

$$\begin{array}{ccc} 0 & \to & \text{Moe} \\ 1 & \to & \text{Larry} \\ 2 & \to & \text{Curly} \end{array}$$

This seems like a bit of a silly thing to do, since we don't need advanced mathematics to tell us that $\{\text{Moe}, \text{Larry}, \text{Curly}\}$ has three elements! But notice three important things about this.

First, a bijection from I_n to another finite set is an awful lot like counting, the way we all learned it when we were little children. Imagine a child sitting on the floor and counting some toys arranged nearby. He will touch each toy and say a single number, starting with 1, then 2, and so on, until the last toy is touched. If the child misses a toy, his older sister, sitting nearby, might say, "But you forgot this one." The child's function from numbers to toys was not surjective, and she's correcting him. Or, if the child comes back around and counts a toy twice, she might point out that he counted it twice, making his function not injective. Even simple counting from childhood is establishing a bijection between the first n numbers and the objects being counted. The only difference here is that we're starting at zero, just to maintain a connection to the structure of $\mathbb{N}$ itself, but we could have chosen to do it differently.

Second, studying counting through the lens of bijections will easily enable us to extend our counting to infinite sets. In fact, you probably dealt with functions among infinite sets (like $f : \mathbb{R} \to \mathbb{R}$ by $f(x) = 2x+7$) earlier in your mathematical education than you did functions among finite sets.

Finally, notice that when there is a bijection in one direction, there is a bijection in the other direction as well. I illustrated above a bijection from I_3 to $\{\text{Moe}, \text{Larry}, \text{Curly}\}$, but if we simply turn each line around, we get a bijection from $\{\text{Moe}, \text{Larry}, \text{Curly}\}$ to I_3.

$$\begin{array}{ccc} \text{Moe} & \to & 0 \\ \text{Larry} & \to & 1 \\ \text{Curly} & \to & 2 \end{array}$$

You can prove that this is always true in the Practice Exercises for this chapter (Theorem 99).

Let us prove the first simple theorem about equinumerosity. You will be asked to prove more theorems in this area in the Practice Exercises.

Theorem 86. *For any set A, we have $A \sim A$.*

Proof. Let A be an arbitrary set. We need to show that there exists a function $f : A \to A$ that is a bijection. I define the function $f : A \to A$ by $f(x) = x$, and I will now show that it is a bijection.

First, I must show that it is injective. By Definition 79, I must show that $\forall x \in A,\ \forall y \in A,\ (f(x) = f(y) \Rightarrow x = y)$. So let x and y be arbitrary elements of A, and assume $f(x) = f(y)$. Then we can conclude $x = y$ by substitution, because we defined f so that $f(x) = x$ and $f(y) = y$.

Second, I must show that it is surjective. By Definition 80, I must show that $\forall y \in A,\ \exists x \in A,\ f(x) = y$. So let y be an arbitrary element of A. I defined f so that $f(y) = y$, and thus the $\exists$ statement is proven.

Thus using Definition 81, I have shown f to be bijective. □

A surprising equinumerosity

Let's see our first somewhat surprising result by taking the tools we've learned so far and applying them to the first infinite case (of many). Consider the even natural numbers, defined as follows.

Definition 87. Let's use E to represent the set of even natural numbers. A natural number n is even if $\exists m \in \mathbb{N},\ 2m = n$.

It turns out that the set of even natural numbers, although it encompasses only half of the natural numbers, is actually the same size as the natural numbers! The following theorem proves this fact.

Theorem 88. $\mathbb{N} \sim E$.

Proof. Let $f : \mathbb{N} \to E$ be given by $f(n) = 2n$. I must prove that f is bijective.

Proof that it is injective: Assume $f(n) = f(m)$ then we have $2n = 2m$ by the definition of f, and then $n = m$ because $2 \neq 0$ and thus it can be cancelled from both sides of the equation (recall Theorem 6 from page 148) yielding $n = m$.

Proof that it is surjective: Let $n \in E$, then by the definition of E we know $\exists m \in \mathbb{N},\ 2m = n$. Thus $\exists m \in \mathbb{N},\ f(m) = n$ by the definition of f and substitution. This is also the definition of surjective (Definition 80), as desired. □

On the one hand, this seems ridiculous. Surely E should only be half as big as $\mathbb{N}$! But on the other hand, it makes sense, because if you pictured E as the list of all even numbers, extending off forever to the right, it has in common one very important thing with $\mathbb{N}$: It is an infinite list of numbers. If we ignore their names (by converting between the sets with a bijection) then they have not only the same size, but even the same structure.

From that point of view, how do you think the size of $\mathbb{N}$ would compare with the size of all natural numbers that are multiples of 3? Or 10? Or 7,000,000?

Quiz Yourself

- Why might someone find the fact $E \sim \mathbb{N}$ surprising?
- Can you explain simply why that equinumerosity makes sense?

14.3 Infinities

What does infinite mean?

I've begun to use the term "infinite" informally. Let's give it an official definition.

Definition 89 (finite, infinite)**.** A set A is *finite* if $\exists n \in \mathbb{N},\ A \sim I_n$. It is *infinite* if it is not finite (i.e., there does not exists such an n).

Theorem 90. *If there is a surjection $f : I_n \to A$ then A is finite.*

The proof of this theorem is surprisingly tricky for how simple it is, so I include this proof here in the chapter. You will be able to leverage this theorem in several of the Practice Exercises for this chapter.

Proof. We prove this result by induction on n.

Base case: If $n = 1$ then $I_n = \{0\}$. We must prove that if $f : I_1 \to A$ is surjective, then A is finite. Notice that $\forall x \in I_1,\ \forall y \in I_1,\ x = y$. Thus surely we can make the even weaker statement $\forall x \in I_1,\ \forall y \in I_1,\ f(x) = f(y) \Rightarrow x = y$; it would require proving a conditional whose assumption we did not even need. Since that is the definition of injective, we find that f itself is also injective, and thus bijective. So f itself proves A to be finite, by Definition 89.

Induction step: Assume that the theorem is true for k. That is, assume that whenever there is a surjection $f : I_k \to A$, then A is finite. We must prove that the theorem is true for $k + 1$, that is, whenever there is a surjection $f : I_{k+1} \to A$, then A is finite. So take an arbitrary surjection $f : I_{k+1} \to A$.

Either f is injective or it is not; we prove the result in both cases. In the case where f is injective, then f is bijective, and thus A is finite by Definition 89, completing the proof. In the case where f is not injective, then we must prove A is finite in some other way. Assuming that f is not injective, we therefore have that $\exists x \in I_{k+1},\ \exists y \in I_{k+1},\ f(x) = f(y) \wedge x \neq y$.

By Theorem 19, either $x < y$, $x > y$, or $x = y$. We know that $x \neq y$, leaving us only two cases. Let's first consider when $x < y$. In that case, I create a function $g : I_k \to A$ by the following piecewise definition.

$$g(m) = \begin{cases} f(m) & \text{if } m < y \\ f(m+1) & \text{if } m \geq y \end{cases}$$

I will now prove that g is surjective. Assume that $a \in A$. Because f is surjective, there is some $m \in I_{k+1}$ such that $f(m) = a$. We proceed in two cases, either $m < y$ or $m \geq y$, by Theorem 16.

If $m < y$ then the definition of g tells us that $g(m) = f(m) = a$. And because $m < y < k+1$, we know that $m < k$. So in this case $\exists m \in I_k,\ g(m) = a$.

If $m \geq y$ then the definition of g tells us that $g(m-1) = f(m) = a$. And because $m \leq y < k+1$, we know that $m - 1 < k$. So in this case again, g maps something from I_k to a.

In all cases we therefore have $\exists m \in I_k,\ g(m) = a$, making g surjective by Definition 80. Using the Indiction Hypothesis, because there is a surjection from I_k to A, A is finite, which is what we wanted to prove. Thus the case where $x < y$ is complete.

To do the proof for the case when $x > y$, simply do the exact same steps, but swapping the roles of x and y. The proof is otherwise identical.

Thus in every case, we have that A is finite, and the induction step is complete. This completes the proof of the whole theorem by mathematical induction. □

Because that proof was rather arduous, we will skip the proof of the following theorem.

Theorem 91. *There is no surjection from any I_n to $\mathbb{N}$, and thus $\mathbb{N}$ is not finite.*

The proof would also be lengthy, and by induction. You may feel free to prove it for yourself, of course, but it is not one of the Practice Exercises. You may, however, cite this theorem in your work in the Practice Exercises.

The final and most interesting results of counting infinite sets arise when we have a way to construct larger and larger infinite sets. The powerset operation enables us to do so, and we define it here.

Powersets

Definition 92. For any set A, the *powerset of* A, written $\mathcal{P}(A)$, is the set of all subsets of A

When listing all the subsets of a finite set, it helps to ask questions like these:

- ▷ What are the subsets of size zero?
- ▷ What are the subsets of size one?
- ▷ What are the subsets of size two?
- ▷ Continue this process up to the size of the whole set.

For example, let's say we wanted to compute $\mathcal{P}\big(\{1,2\}\big)$. Asking what subsets are of size zero yields only one, the empty set, $\{\}$. Asking what subsets are of size one yields two subsets, $\{1\}$ and $\{2\}$. Asking what subsets are of size two yields only the whole set in question, $\{1,2\}$. Thus we have found four subsets of $\{1,2\}$, and express our answer as follows.

$$\mathcal{P}\big(\{1,2\}\big) = \Big\{\{\},\{1\},\{2\},\{1,2\}\Big\}$$

Notice that the powerset is a *set of sets.* We took the four subsets we found and placed them *in another set* to form the powerset. I use two different size braces in the expression above to make this clear.

For larger finite sets, the powerset is significantly larger, but it can be computed by the same step-by-step technique. But regardless of whether the initial set A is finite or infinite, Georg Cantor proved the following fascinating result about the relative sizes of A and $\mathcal{P}(A)$.

Theorem 93 (Cantor)**.** *For any set A, there does not exist a surjection $f : A \to \mathcal{P}(A)$.*

Proof. Assume towards a contradiction that $\exists f : A \to \mathcal{P}(A)$, and f is a surjection. Consider the set $S = \{x \in A \mid x \notin f(x)\}$.

This set S is clearly a subset of A, because it's defined to be all the $x \in A$ that meet a certain condition. So we know that $S \subseteq A$. Because the definition of $\mathcal{P}(A)$ says that it's the set of *all* subsets of A, we therefore know that $S \in \mathcal{P}(A)$. Putting this together with the fact that $f : A \to \mathcal{P}(A)$ is a surjection, the definition of surjection tells us that $\exists a \in A,\ f(a) = S$.

Assume towards a contradiction that $a \in S$. Recall the definition of S, above. Together with our assumption, that lets us conclude that $a \notin f(a)$. But since $f(a) = S$, we can substitute into that statement to obtain $a \notin S$. That's a contradiction with our assumption $a \in S$, therefore disproving our assumption.

Thus we have proven $a \notin S$. Again, using the definition of S, above, that means that $a \in f(a)$. Using substitution again with $f(a) = S$, we have $a \in S$. This is a contradiction with the fact we just proved, $a \notin S$. Thus the original assumption at the top of our proof is invalid; there cannot exists a surjection $f : A \to \mathcal{P}(A)$. This proves the theorem. □

This theorem says that *every* powerset is *bigger* than the set it was computed from. So if you ever need to create a set that's bigger than one you've got, just apply $\mathcal{P}$ to it. This gives us the following surprising result.

Theorem 94. *There are infinitely many different sizes of infinite sets.*

Proof. Start with $\mathbb{N}$, which is infinite. Then $\mathcal{P}(\mathbb{N})$ is another infinite set, but is strictly bigger than $\mathbb{N}$. And $\mathcal{P}(\mathcal{P}(\mathbb{N}))$ is another infinite set, strictly bigger still. And $\mathcal{P}(\mathcal{P}(\mathcal{P}(\mathbb{N})))$ is another infinite set, strictly bigger still. And so on. □

It might have been surprising enough to find out that not all infinite sets are the same size. But we just found out that there are *infinitely many* different size of infinite sets!

Relative sizes of $\mathbb{N}$ and $\mathbb{R}$

The following theorem is very helpful in creating bijections.

Theorem 95 (Schröder-Bernstein)**.** *If there is an injection from A to B and another injection from B to A, then there is a bijection between them.*

Thus, if you need to prove that there is a bijection from A to B (that is, that $A \sim B$), you can create two separate injections, one in each direction. These two injections need not necessarily have anything to do with one another, and thus your overall job is much easier than trying to create one unified bijection.

The proof of the Schröder-Bernstein Theorem takes a few pages, and thus I do not include it here. Although you have the mathematical and logical background now to understand the proof, I do not want to distract us from the use to which we will put that theorem.

We are about to prove that the real numbers $\mathbb{R}$ are a much larger collection than the natural numbers $\mathbb{N}$. Theorem 93 already showed that $\mathcal{P}(\mathbb{N})$ is strictly larger than $\mathbb{N}$, and we will now prove that $\mathcal{P}(\mathbb{N}) \sim \mathbb{R}$, thereby proving that $\mathbb{R}$, too, is larger than $\mathbb{N}$.

Theorem 96. $\mathcal{P}(\mathbb{N}) \sim \mathbb{R}$

Proof. First, I create an injection $f : \mathcal{P}(\mathbb{N}) \to \mathbb{R}$. Let $f : \mathcal{P}(\mathbb{N}) \to \mathbb{R}$ be defined as follows. On an input set $S \subseteq \mathbb{N}$, create an output as follows. Take the numbers in S and write them in increasing order, $n_1, n_2, n_3, \ldots$. Convert each one to a long string of ones; thus n_1 would become a string of n_1 ones, and n_2 becomes n_2 ones. (For example, 5 would turn into 11111.) Create a decimal by stringing together all those binary representations, and placing a zero after each. That decimal is the real number output of $f(S)$.

For example, let's compute $f(\{5, 10\})$. Converting to strings of ones, we treat $\{5, 10\}$ as $\{11111, 1111111111\}$. To find $f(S)$, we therefore place 11111 and 1111111111 in sequence, with a 0 after each, forming the decimal

$$0.11111011111111110.$$

It should be clear that this function is injective for the following reason. If $f(S_1) = f(S_2)$, then we can tell exactly what numbers are in the set S_1 by reading down the decimal expansion of $f(S_1)$. It will tell us every single natural number in S_1, in increasing order. Just find a sequence of ones (stopping when you see a zero) and count how many ones there were, then proceed on to the next block of ones. Since that same decimal expansion tells us the numbers in S_2, we will find that they are the exact same sets.

Thus we have an injection from $\mathcal{P}(\mathbb{N})$ to $\mathbb{R}$. Now I create an injection $g : \mathbb{R} \to \mathcal{P}(\mathbb{N})$. Again, we will use longs strings of digits to spell out information. A real number $r \in \mathbb{R}$ has a whole number to the left of the decimal place, followed by potentially infinitely many digits to the right of the decimal place. We wish to define a set S that is the output of $g(r)$; I do so as follows.

Take the number to the left of the decimal place in r, and encode it by writing that many ones in a row. So if $r = \pi = 3.14159165358979\ldots$, the number to the left of the decimal point is 3, and we thus encode it as 111. Place that number in S. In order to distinguish positive numbers from negative numbers, if r is negative, place a 2 in front of the sequence of ones, to act as a minus sign. (So for example if $r = -\pi$, we would place 2111 into S.)

As for the decimal places in r, encode each one as follows. For the digit d that sits n digits to the right of the decimal place, write d threes followed by n fours. So for example, we would proceed with $r = \pi = 3.14159265358979\ldots$ as follows. The 1 that is 1 digit to the right of the decimal becomes 34, and 34 is placed in S. The 4 that is 2 digits to the right of the decimal becomes 333344, which goes in S. The 1 that is 3 digits to the right of the decimal becomes 3444, which is placed in S, and so on, for a finite or infinite decimal.

This function, too, is injective, because one can use S to read exactly what the initial real number r was. So if we have $g(r_1) = g(r_2)$, we know that $r_1 = r_2$ as follows. First, look inside S and find the only number that has ones in it. It will tell you the value to the left of the decimal, and whether it is positive or negative. Then take out every other number from S one at a time, using them to fill in the places to the right of the decimal. Although this process may have infinitely many steps, when you're done, you know r_1 perfectly. And if you repeated the process to find r_2, you'd find that it was the exact same number. Thus there is an injection $g : \mathbb{R} \to \mathcal{P}(\mathbb{N})$.

By Theorem 95, since we have injections in both directions between $\mathbb{N}$ and $\mathcal{P}(\mathbb{R})$, they are the same size. $\square$

You have now seen some new mathematical vistas! There are infinitely many sizes of infinite sets, and two of the *smallest* ones are $\mathbb{N}$ and $\mathbb{R}$. You will find out the relative sizes of $\mathbb{Z}$ and $\mathbb{Q}$ compared to $\mathbb{N}$, $\mathbb{R}$, and one another, in the following Practice Exercises.

Quiz Yourself

- Which of the theorems in the previous section was proven by *reductio,* and how can you tell?
- Using the function f from Theorem 96, compute $f(\{1,2,3\})$.
- For the same f, what set $S \subseteq \mathbb{N}$ satisfies $f(S) = 0.11011111$?
- Using the function g from Theorem 96, compute $g(6.2501)$.
- For the same g, what number x satisfies $g(x) = \{21111111, 333333444\}$?

Practice Exercises

Part A

How would you phrase Theorem 77 differently, given the terminology defined in Section 14.1?

How would you phrase Theorem 78 differently, given the terminology defined in Section 14.1?

Part B

Let $f : \{1,2,3\} \to \{4,5,6\}$ be given by $f(x) = x + 3$.

Let $g : \{4,5,6\} \to \{7,8,9\}$ be given by $g(x) = 13 - x$.

Draw a picture representing the function $f \circ g$.

Let $h : \{a,b\} \to \{1,2,3\}$ be given by $h(a) = 3$ and $h(b) = 1$.

Let $k : \{1,2,3\} \to \{1,2,3\}$ be given by $k(x) = (x-2)^2 + 1$.

Draw a picture representing the function $h \circ k$.

Part C

Come up with a bijection between $\mathbb{N}$ and $\{1,2,3,4,\ldots\}$.

Part D

For some $n \in \mathbb{N}$ of your choosing, come up with an example surjection $f : I_n \to \{\text{parsley}, \text{sage}, \text{rosemary}, \text{bacon}\}$ that is *not* also an injection.

Part E

Let's say a natural number n is odd if $n - 1$ is even. In other words,

$$n \in O \quad \Leftrightarrow \quad n - 1 \in E.$$

How would you change the proof of Theorem 88 to show that $\mathbb{N} \sim O$?

Part F

Choose any *one* of the following theorems to prove.

Theorem 97. *The composition of two surjections is a surjection.*

Theorem 98. *The composition of two bijections is a bijection.*

Theorem 99. *If $f : A \to B$ is a bijection then there is some other bijection $g : B \to A$ such that $\forall x \in A,\ g(f(x)) = x$ and $\forall y \in B,\ f(g(y)) = y$.*

(And what is such a bijection typically called?)

Theorem 100. *Let $f : \mathbb{R} \to \mathbb{R}$ be $f(x) = ax + b$ for $a, b \in \mathbb{R}$ and $a \neq 0$. Then f is a bijection.*

Part G

Choose any *one* of the following theorems to prove.

Theorem 101. *If $A \sim B$ then $B \sim A$.*

Theorem 102. *If $A \sim B$ and $B \sim C$ then $A \sim C$.*

Both of these are easier to prove than $A \sim A$. Hint: Utilize earlier theorems.

Part H

Choose any *one* of the following theorems, and rather than giving a full proof, define a bijection between the two sets. You do not need to write the proof that it is a bijection, but be sure that you could if you had to, so that you know your function *is* a bijection.

Theorem 103. $\mathbb{N} \sim \mathbb{Z}$

Theorem 104. $\mathbb{R} \sim$ *the set of all real numbers except zero*

Theorem 105. $\mathbb{R} \sim$ *the set of positive real numbers*

Part I

Choose any *one* of the following theorems to prove.

Theorem 106. *If A and B are finite then $A \cup B$ is finite.*

Theorem 107. *If A and B are finite then $A \cap B$ is finite.*

Part J

1. Compute $\mathcal{P}(\{1\})$
2. List a few example elements of $\mathcal{P}(\mathbb{N})$.
3. Give an example of something in $\mathcal{P}(\mathbb{R})$ that's not in $\mathcal{P}(\mathbb{N})$.
4. Compute $\mathcal{P}(\emptyset)$.

Part K

Choose any *one* of the following theorems, and rather than giving a full proof, define two injections between the two sets, one in each direction. You do not need to write the proofs that each are injections, but be sure that you could if you had to, so that you know your functions *are* injections. Then apply the Schröder-Bernstein Theorem (Theorem 95) to conclude that there is a bijection between the two sets.

Theorem 108. *Consider the set P of points (x, y) in the plane with both x and y in $\mathbb{N}$. (E.g., $(1, 2)$ and $(0, 7) \in P$, but $(0, 0.5)$ and $(-1, 1) \notin P$.) Then $\mathbb{N} \sim P$.*

Theorem 109. *Same as the previous, but with x and y in $\mathbb{Z}$ instead of just $\mathbb{N}$.*

Theorem 110. *Consider the set F of all functions $f : \mathbb{N} \to \mathbb{N}$.
(E.g., $f(n) = 2n + 7$ is in F, but $g(n) = \frac{n}{2}$ is not.) Then $\mathbb{R} \sim F$.*

Theorem 111. $\mathbb{N} \sim \mathbb{Q}$

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 symbols used for various logical constants is arbitrary. There is nothing written in heaven that says that '$\neg$' 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 '$\neg$' 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.

negation	$\neg$, $\sim$
conjunction	&, $\wedge$, $\bullet$
disjunction	$\vee$, $+$
conditional	$\rightarrow$, $\Rightarrow$, $\supset$
biconditional	$\leftrightarrow$, $\Leftrightarrow$, $\equiv$

Negation Two commonly used symbols are the *hoe*, '$\neg$', and the *swung dash*, '$\sim$.' In some more advanced formal systems it is necessary to distinguish between two kinds of negation; the distinction is sometimes represented by using both '$\neg$' and '$\sim$.'

Disjunction The symbol '$\vee$' 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, '$\wedge$' is a counterpart to the symbol used for disjunction. Sometimes a single dot, '$\bullet$', 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*, '$\rightarrow$', and the *hook*, '$\supset$.'

Material Biconditional The *double-headed arrow*, ‘$\leftrightarrow$’, 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*, ‘$\equiv$’, for the biconditional.

Quantifiers The universal quantifier is typically symbolized as an upside-down A, ‘$\forall$’, and the existential quantifier as a backwards E, ‘$\exists$.’ 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 $\exists xPx$, 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 ‘$\bigvee$’ instead of ‘$\exists$.’

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.

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.)

notation of SL	Polish notation
$\neg$	N
$\wedge$	K
$\vee$	A
$\Rightarrow$	C
$\Leftrightarrow$	E

In Polish notation, a binary connective is written *before* the two sentences that it connects. For example, the sentence $A \wedge B$ of SL would be written Kab in Polish notation.

The sentences $\neg A \Rightarrow B$ and $\neg(A \Rightarrow B)$ 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 $\neg(A \wedge B)$ 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 2 Part C

1. consistent
2. inconsistent
3. consistent
4. consistent

Chapter 2 Part D 1, 2, 3, 6, 8, and 10 are possible.

Chapter 3 Part A

1. $\neg M$
2. $M \vee \neg M$
3. $G \vee C$
4. $\neg C \wedge \neg G$
5. $C \Rightarrow (\neg G \wedge \neg M)$
6. $M \vee (C \vee G)$

Chapter 3 Part C

1. $AE \wedge HE$
2. $AF \Rightarrow AS$
3. $AF \vee AE$
4. $HE \wedge \neg HS$
5. $\neg AE \wedge \neg HE$
6. $AE \wedge HE \wedge \neg(AS \vee HS)$
7. $HS \Rightarrow HF$
8. $(\neg AE \Rightarrow \neg HE) \wedge (AE \Rightarrow HE)$
9. $AS \Leftrightarrow \neg HS$
10. $(HE \wedge HF) \Rightarrow HS$
11. $\neg(HE \wedge HF)$
12. $(AF \wedge HF) \Leftrightarrow (\neg AE \wedge \neg HE)$

Chapter 3 Part E

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. $A \wedge B$
2. $(A \vee B) \Rightarrow C$
3. $\neg(A \vee B) \Rightarrow \neg C$
4. $G \vee C$
5. $(C \vee \neg C) \wedge G$
6. $(A \vee B) \wedge \neg(A \wedge B)$

Chapter 3 Part H

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 4 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 4 Part B 2, 3, 5, 6, 8, and 9 are logically equivalent.

Chapter 4 Part C 1, 3, 6, 7, and 8 are consistent.

Chapter 4 Part D 3, 5, 8, and 10 are valid.

Chapter 4 Part E

1. $\mathcal{A}$ and $\mathcal{B}$ have the same truth value on every line of a complete truth table, so $\mathcal{A} \Leftrightarrow \mathcal{B}$ 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, $\mathcal{A}$ and $\mathcal{B}$ are true and $\mathcal{C}$ 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 $\mathcal{A}$ is false on every line of a complete truth table, there is no line on which $\mathcal{A}$ and $\mathcal{B}$ are true and $\mathcal{C}$ is false. So the argument is valid.
5. Since $\mathcal{C}$ is true on every line of a complete truth table, there is no line on which $\mathcal{A}$ and $\mathcal{B}$ are true and $\mathcal{C}$ is false. So the argument is valid.
6. Not much. $(\mathcal{A} \vee \mathcal{B})$ is a tautology if $\mathcal{A}$ and $\mathcal{B}$ are tautologies; it is a contradiction if they are contradictions; it is contingent if they are contingent.
7. $\mathcal{A}$ and $\mathcal{B}$ have different truth values on at least one line of a complete truth table, and $(\mathcal{A} \vee \mathcal{B})$ will be true on that line. On other lines, it might be true or false. So $(\mathcal{A} \vee \mathcal{B})$ is either a tautology or it is contingent; it is *not* a contradiction.

Chapter 4 Part F

1. $\neg A \Rightarrow B$
2. $\neg(A \Rightarrow \neg B)$
3. $\neg((A \Rightarrow B) \Rightarrow \neg(B \Rightarrow A))$

Chapter 7 Part A

1. $zoo(A) \wedge zoo(B) \wedge zoo(C)$
2. $rep(B) \wedge \neg alli(B)$
3. $loves(C, B) \Rightarrow mon(B)$
4. $(alli(B) \wedge alli(C)) \Rightarrow (loves(A, B) \wedge loves(A, C))$
5. $\exists x, (rep(x) \wedge zoo(x))$
6. $\forall x, (alli(x) \Rightarrow rep(x))$
7. $\forall x, \big(zoo(x) \Rightarrow (mon(x) \vee alli(x))\big)$
8. $\exists x, (rep(x) \wedge \neg alli(x))$
9. $\exists x, (rep(x) \wedge loves(C, x))$
10. $\forall x, \big((mon(x) \wedge zoo(x)) \Rightarrow loves(B, x)\big)$
11. $\forall x, \big[(mon(x) \wedge loves(A, x)) \Rightarrow loves(x, A)\big]$
12. $\exists x, rep(x) \Rightarrow rep(A)$
13. $\forall x, (alli(x) \Rightarrow rep(x))$
14. $\forall x, \big((mon(x) \wedge loves(C, x)) \Rightarrow loves(A, x)\big)$
15. $\exists x, (mon(x) \wedge loves(x, B) \wedge \neg loves(B, x))$

Chapter 7 Part E

1. $\neg\exists x, tried(x)$
2. $\forall x, (mar(x) \Rightarrow sugar(x))$
3. $\exists x, \neg sugar(x)$
4. $\exists x, [choc(x) \wedge \neg\exists y, better(y, x)]$
5. $\neg\exists x, better(x, x)$
6. $\neg\exists x, (choc(x) \wedge \neg sugar(x) \wedge tried(x))$
7. $\exists x, (choc(x) \wedge tried(x)) \wedge \exists x, (mar(x) \wedge tried(x)) \wedge \neg\exists x, (choc(x) \wedge mar(x) \wedge tried(x))$
8. $\forall x, (choc(x) \Rightarrow \forall y, (\neg choc(y) \Rightarrow better(x, y)))$
9. $\forall x, \big((choc(x) \wedge mar(x)) \Rightarrow \forall y, [(\neg choc(y) \wedge \neg mar(y)) \Rightarrow better(x, y)]\big)$

Chapter 7 Part G

1. $\forall x, (child(x, P) \Rightarrow dance(x))$
2. $child(J, P) \wedge fem(J)$
3. $\exists x, (child(x, P) \wedge fem(x))$
4. $\neg\exists x, sib(x, J)$
5. $\forall x, \big((child(x, P) \wedge fem(x)) \Rightarrow dance(x)\big)$
6. $\neg\exists x, (child(x, P) \wedge male(x))$
7. $\exists x, (child(J, x) \wedge sib(x, E) \wedge fem(J))$
8. $sib(P, E) \wedge male(P)$
9. $\forall x, \big((sib(x, P) \wedge male(x)) \Rightarrow \neg\exists y, child(y, x)\big)$
10. $\exists x, (sib(x, J) \wedge \exists y, child(y, x) \wedge fem(J))$
11. $\forall x, \big(dance(x) \Rightarrow \exists y, (sib(x, y) \wedge fem(y) \wedge dance(y))\big)$
12. $\forall x, \big((male(x) \wedge dance(x)) \Rightarrow \exists y, (child(x, y) \wedge dance(y))\big)$

Chapter 7 Part I

1. $related(C, A)$, $related(C, B)$, $related(C, C)$, and $related(C, D)$ are substitution instances of $\forall x, related(C, x)$.
2. Of the expressions listed, only $\forall y, likes(B, y)$ is a substitution instance of $\exists x, \forall y, likes(x, y)$.

Chapter 7 Part K

1. $\forall x, (club(x) \Rightarrow black(x))$
2. $\neg\exists x, wild(x)$
3. $\exists x, \exists y, (club(x) \wedge club(y) \wedge x \neq y)$
4. $\exists x, \exists y, (jack(x) \wedge eye(x) \wedge jack(y) \wedge eye(y) \wedge x \neq y)$
5. $\forall x, \forall y, \forall z, \big((jack(x) \wedge eye(x) \wedge jack(y) \wedge eye(y) \wedge jack(z) \wedge eye(z)) \Rightarrow (x = y \vee x = z \vee y = z)\big)$
6. $\exists x, \exists y, \big(jack(x) \wedge black(x) \wedge jack(y) \wedge black(y) \wedge x \neq y \wedge \forall z, ((jack(z) \wedge black(z)) \Rightarrow (x = z \vee y = z))\big)$
7. $\exists a, \exists b, \exists c, \exists d, \big(deuce(a) \wedge deuce(b) \wedge deuce(c) \wedge deuce(d) \wedge a \neq b \wedge a \neq c \wedge a \neq d \wedge b \neq c \wedge b \neq d \wedge c \neq d \wedge \neg\exists y, (deuce(y) \wedge y \neq a \wedge y \neq b \wedge y \neq c \wedge y \neq d)\big)$
8. $\exists x, \big(deuce(x) \wedge club(x) \wedge \forall y, [(deuce(y) \wedge club(y)) \Rightarrow x = y] \wedge black(x)\big)$
9. $\forall x, \big((eye(x) \wedge jack(x)) \Rightarrow wild(x)\big) \wedge \exists x, \big(axe(x) \wedge \forall y, (axe(y) \Rightarrow x = y) \wedge wild(x)\big)$
10. $\exists x, \big(deuce(x) \wedge club(x) \wedge \forall y, ((deuce(y) \wedge club(y)) \Rightarrow x = y) \wedge wild(x)\big) \Rightarrow \exists x, \forall y, (wild(x) \Leftrightarrow x = y)$
11. wide scope: $\neg\exists x, \big(axe(x) \wedge \forall y, (axe(y) \Rightarrow x = y) \wedge jack(x)\big)$
    narrow scope: $\exists x, \big(axe(x) \wedge \forall y, (axe(y) \Rightarrow x = y) \wedge \neg jack(x)\big)$
12. wide scope: $\neg\exists x, \exists z, \big(deuce(x) \wedge club(x) \wedge axe(z) \wedge \forall y, ((deuce(y) \wedge club(y)) \Rightarrow x = y) \wedge \forall y, ((axe(y) \Rightarrow z = y) \wedge x = z)\big)$
    narrow scope: $\exists x, \exists z, \big(deuce(x) \wedge club(x) \wedge axe(z) \wedge \forall y, ((deuce(y) \wedge club(y)) \Rightarrow x = y) \wedge \forall y, ((axe(y) \Rightarrow z = y) \wedge x \neq z)\big)$

Chapter 8 Part A 2, 3, 4, 6, 8, and 9 are true in the model.

Chapter 8 Part B 4, 5, and 7 are true in the model.

Chapter 8 Part D

$$\begin{aligned} \text{UD} &= \{10,11,12,13\} \\ \text{extension}(odd) &= \{11,13\} \\ \text{extension}(ls) &= \emptyset \\ \text{extension}(td) &= \{10,11,12,13\} \\ \text{extension}(tu) &= \{13\} \\ \text{extension}(nna) &= \{(11,10),(12,11),(13,12)\} \end{aligned}$$

Chapter 8 Part E

1. The sentence is true in this model:

$$\begin{aligned}\text{UD} &= \{\text{Stan}\}\\ \text{extension}(\textit{dog}) &= \{\text{Stan}\}\\ \text{referent}(A) &= \text{Stan}\\ \text{referent}(B) &= \text{Stan}\end{aligned}$$

   And it is false in this model:

$$\begin{aligned}\text{UD} &= \{\text{Stan}\}\\ \text{extension}(\textit{dog}) &= \emptyset\\ \text{referent}(A) &= \text{Stan}\\ \text{referent}(B) &= \text{Stan}\end{aligned}$$

2. The sentence is true in this model:

$$\begin{aligned}\text{UD} &= \{\text{Stan}\}\\ \text{extension}(\textit{tall}) &= \{(\text{Stan, Stan})\}\\ \text{referent}(H) &= \text{Stan}\end{aligned}$$

   And it is false in this model:

$$\begin{aligned}\text{UD} &= \{\text{Stan}\}\\ \text{extension}(\textit{tall}) &= \emptyset\\ \text{referent}(H) &= \text{Stan}\end{aligned}$$

3. The sentence is true in this model:

$$\begin{aligned}\text{UD} &= \{\text{Stan, Ollie}\}\\ \text{extension}(\textit{pretty}) &= \{\text{Stan}\}\\ \text{referent}(M) &= \text{Stan}\end{aligned}$$

   And it is false in this model:

$$\begin{aligned}\text{UD} &= \{\text{Stan}\}\\ \text{extension}(\textit{pretty}) &= \emptyset\\ \text{referent}(M) &= \text{Stan}\end{aligned}$$

Chapter 8 Part F There are many possible correct answers. Here are some:

1. Making the first sentence true and the second false:

$$\begin{aligned}\text{UD} &= \{\text{alpha}\}\\ \text{extension}(\textit{joking}) &= \{\text{alpha}\}\\ \text{extension}(\textit{kidding}) &= \emptyset\\ \text{referent}(A) &= \text{alpha}\end{aligned}$$

2. Making the first sentence true and the second false:

$$\begin{aligned}\text{UD} &= \{\text{alpha, omega}\}\\ \text{extension}(\textit{joking}) &= \{\text{alpha}\}\\ \text{referent}(M) &= \text{omega}\end{aligned}$$

3. Making the first sentence false and the second true:

$$\begin{aligned}\text{UD} &= \{\text{alpha, omega}\}\\ \text{extension}(\textit{related}) &= \{(\text{alpha,alpha})\}\end{aligned}$$

4. Making the first sentence false and the second true:

$$\begin{aligned}\text{UD} &= \{\text{alpha, omega}\}\\ \text{extension}(\textit{pretty}) &= \{\text{alpha}\}\\ \text{extension}(\textit{quiet}) &= \emptyset\\ \text{referent}(C) &= \text{alpha}\end{aligned}$$

5. Making the first sentence true and the second false:

$$\begin{aligned}\text{UD} &= \{\text{iota}\}\\ \text{extension}(\textit{pretty}) &= \emptyset\\ \text{extension}(\textit{quiet}) &= \emptyset\end{aligned}$$

6. Making the first sentence false and the second true:
$$\begin{aligned}\text{UD} &= \{\text{iota}\}\\ \text{extension}(\textit{pretty}) &= \emptyset\\ \text{extension}(\textit{quiet}) &= \{\text{iota}\}\end{aligned}$$
7. Making the first sentence true and the second false:
$$\begin{aligned}\text{UD} &= \{\text{iota}\}\\ \text{extension}(\textit{pretty}) &= \emptyset\\ \text{extension}(\textit{quiet}) &= \{\text{iota}\}\end{aligned}$$
8. Making the first sentence true and the second false:
$$\begin{aligned}\text{UD} &= \{\text{alpha, omega}\}\\ \text{extension}(\textit{related}) &= \{(\text{alpha, omega}), (\text{omega, alpha})\}\end{aligned}$$
9. Making the first sentence false and the second true:
$$\begin{aligned}\text{UD} &= \{\text{alpha, omega}\}\\ \text{extension}(\textit{related}) &= \{(\text{alpha, alpha}), (\text{alpha, omega})\}\end{aligned}$$

Chapter 8 Part I

1. There are many possible answers. Here is one:
$$\begin{aligned}\text{UD} &= \{\text{Harry, Sally}\}\\ \text{extension}(\textit{related}) &= \{(\text{Sally, Harry})\}\\ \text{referent}(A) &= \text{Harry}\end{aligned}$$
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 referent$(A) \neq \beta$. But if referent$(A) \neq \beta$, 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.

Chapter 8 Part J

2. No, it would not make any difference. The satisfaction of a sentence 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 9 Part A

1.	$\forall x, \exists y, (related(x, y) \vee related(y, x))$	given
2.	$\forall x, \neg related(M, x)$	given
3.	$\exists y, (related(M, y) \vee related(y, M))$	$\forall$E 1.
4.	Let A be such that:	
5.	$related(M, A) \vee related(A, M)$	$\exists$E 3.
6.	$\neg related(M, A)$	$\forall$E 2.
7.	$related(A, M)$	$\vee$E 5., 6.
8.	$\exists x, related(x, M)$	$\exists$I 7.

1.	$\forall x, (\exists y, less(x,y) \Rightarrow \forall z, less(z,x))$	given
2.	$less(A,B)$	given
3.	$\exists y, less(A,y) \Rightarrow \forall z, less(z,A)$	∀E 1.
4.	$\exists y, less(A,y)$	∃I 2.
5.	$\forall z, less(z,A)$	⇒E 3., 4.
6.	Let c be arbitrary.	
7.	$less(c,A)$	∀E 5.
8.	$\exists y, less(c,y) \Rightarrow \forall z, less(z,c)$	∀E 1.
9.	$\exists y, less(c,y)$	∃I 7.
10.	$\forall z, less(z,c)$	⇒E 8., 9.
11.	$less(c,c)$	∀E 10.
12.	$\forall x, less(x,x)$	∀I 6., 11.

1.	$\forall x, (jog(x) \Rightarrow knit(x))$	given
2.	$\exists x, \forall y, less(x,y)$	given
3.	$\forall x, jog(x)$	given
4.	Let A be such that:	
5.	$\forall y, less(A,y)$	∃E 2., 4.
6.	$less(A,A)$	∀E 5.
7.	$jog(A)$	∀E 3.
8.	$jog(A) \Rightarrow knit(A)$	∀E 1.
9.	$knit(A)$	⇒E 7., 8.
10.	$knit(A) \wedge less(A,A)$	∧I 9., 6.
11.	$\exists x, (knit(x) \wedge less(x,x))$	∃I 10.

1.	$\quad\neg(\exists x, man(x) \vee \forall x, \neg man(x))$	
2.	$\quad\neg\exists x, man(x) \wedge \neg\forall x, \neg man(x)$	DeM 1.
3.	$\quad\neg\exists x, man(x)$	∧E 2.
4.	$\quad\forall x, \neg man(x)$	QN 3.
5.	$\quad\neg\forall x, \neg man(x)$	∧E 2.
6.	$\exists x, man(x) \vee \forall x, \neg man(x)$	¬E 1., 4., 5.

Chapter 9 Part B

1.

1.	$\neg(\forall x, fly(x) \lor \neg\forall x, fly(x))$	for reductio
2.	$\neg\forall x, fly(x) \land \neg\neg\forall x, fly(x)$	DeM 1.
3.	$\neg\forall x, fly(x)$	$\land$E 2.
4.	$\neg\neg\forall x, fly(x)$	$\land$E 2.
5.	$\forall x, fly(x) \lor \neg\forall x, fly(x)$	$\neg$E 1., 3., 4.

2.

1.	$\forall x, (man(x) \Leftrightarrow nice(x))$	
2.	$man(A) \land \exists x, related(x, A)$	want $\exists x, nice(x)$
3.	$man(A) \Leftrightarrow nice(A)$	$\forall$E 1.
4.	$man(A)$	$\land$E 2.
5.	$nice(A)$	$\Leftrightarrow$E 3., 4.
6.	$\exists x, nice(x)$	$\exists$I 5.

3.

1.	$\forall x, (\neg man(x) \lor less(J, x))$	given
2.	$\forall x, (bad(x) \Rightarrow less(J, x))$	given
3.	$\forall x, (man(x) \lor bad(x))$	given; want $\forall x, less(J, x)$
4.	Let a be arbitrary.	
5.	$\neg man(a) \lor less(J, a)$	$\forall$E 1.
6.	$man(a) \Rightarrow less(J, a)$	MC 5.
7.	$bad(a) \Rightarrow less(J, a)$	$\forall$E 2.
8.	$man(a) \lor bad(a)$	$\forall$E 3.
9.	$less(J, a)$	$\lor *$ 8., 6., 7.
10.	$\forall x, less(J, x)$	$\forall$I 4., 9.

4.

1.	$\forall x, (cute(x) \land dog(T))$	given; want $\forall x, cute(x) \land dog(T)$
2.	Let a be arbitrary.	
3.	$cute(a) \land dog(T)$	$\forall$E 1.
4.	$cute(a)$	$\land$E 3.
5.	$\forall x, cute(x)$	$\forall$I 2., 4.
6.	$dog(T)$	$\land$E 3.
7.	$\forall x, cute(x) \land dog(T)$	$\land$I 5., 6.

5.

1.	$\exists x, (cute(x) \vee dog(T))$	given; want $\exists x, cute(x) \vee dog(T)$
2.	Let A be such that:	
3.	$cute(A) \vee dog(T)$	$\exists$E 1., 2.
4.	$\quad \neg(\exists x, cute(x) \vee dog(T))$	for reductio
5.	$\quad \neg\exists x, cute(x) \wedge \neg dog(T)$	DeM 4.
6.	$\quad \neg dog(T)$	$\wedge$E 5.
7.	$\quad cute(A)$	$\vee$E 3., 6.
8.	$\quad \exists x, cute(x)$	$\exists$I 7.
9.	$\quad \neg\exists x, cute(x)$	$\wedge$E 5.
10.	$\exists x, cute(x) \vee dog(T)$	$\neg$E 4., 8., 9.

Chapter 9 Part I Regarding the translation of this argument, see p. 89.

1.	$\exists x, \forall y, \big(\forall z, (likes(x,z) \Rightarrow likes(y,z)) \Rightarrow likes(x,y)\big)$	given
2.	Let A be such that:	
3.	$\forall y, \big(\forall z, (likes(A,z) \Rightarrow likes(y,z)) \Rightarrow likes(A,y)\big)$	$\exists$E 1., 2.
4.	$\forall z, (likes(A,z) \Rightarrow likes(A,z)) \Rightarrow likes(A,A)$	$\forall$E 3.
5.	$\quad \neg\exists x, likes(x,x)$	for reductio
6.	$\quad \forall x, \neg likes(x,x)$	QN 5.
7.	$\quad \neg likes(A,A)$	$\forall$E 6.
8.	$\quad \neg\forall z, (likes(A,z) \Rightarrow likes(A,z))$	MT 4., 7.
9.	$\quad\quad$ Let b be arbitrary.	
10.	$\quad\quad\quad likes(A,b)$	
11.	$\quad\quad\quad likes(A,b)$	R 10.
12.	$\quad\quad likes(A,b) \Rightarrow likes(A,b)$	$\Rightarrow$I 10., 11.
13.	$\quad \forall z, (likes(A,z) \Rightarrow likes(A,z))$	$\forall$I 9., 12.
14.	$\exists x, likes(x,x)$	$\neg$E 5., 13., 8.

Chapter 9 Part M 2, 3, and 5 are logically equivalent.

Chapter 9 Part N 2, 4, 5, 7, and 10 are valid. Here are complete answers for some of them:

1.

$$\begin{aligned} \text{UD} &= \{\text{mocha, freddo}\} \\ \text{extension}(\textit{related}) &= \{(\text{mocha, freddo}), (\text{freddo, mocha})\} \end{aligned}$$

2.

1.	$\exists y, \forall x, related(x, y)$	given; want $\forall x, \exists y, related(x, y)$
2.	Let A be such that:	
3.	$\forall x, related(x, A)$	$\exists$E 1., 2.
4.	Let b be arbitrary.	
5.	$related(b, A)$	$\forall$E 3.
6.	$\exists y, related(b, y)$	$\exists$I 5.
7.	$\forall x, \exists y, related(x, y)$	$\forall$I 4., 6.

Quick Reference

Characteristic Truth Tables

$\mathcal{A}$	$\neg\mathcal{A}$
T	F
F	T

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{A}\wedge\mathcal{B}$	$\mathcal{A}\vee\mathcal{B}$	$\mathcal{A}\Rightarrow\mathcal{B}$	$\mathcal{A}\Leftrightarrow\mathcal{B}$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

Symbolization

Sentential Connectives (chapter 3)

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 $(\neg P \wedge \neg Q)$
Both P, and Q.	$(P \wedge Q)$
If P, then Q.	$(P \Rightarrow Q)$
P only if Q.	$(P \Rightarrow Q)$
P if and only if Q.	$(P \Leftrightarrow Q)$
Unless P, Q. P unless Q.	$(P \vee Q)$

Predicates (chapter 7)

All cats are cute.	$\forall x, (cat(x) \Rightarrow cute(x))$
Some cats are cute.	$\exists x, (cat(x) \wedge cute(x))$
Not all cats are cute.	$\neg\forall x, (cat(x) \Rightarrow cute(x))$ or $\exists x, (cat(x) \wedge \neg cute(x))$
No cats are cute.	$\forall x, (cat(x) \Rightarrow \neg cute(x))$ or $\neg\exists x, (cat(x) \wedge cute(x))$

Identity (section 7.6)

Only B is big.	$\forall x, (big(B) \Leftrightarrow x = B)$
Everything besides B is big.	$\forall x, (x \neq B \Rightarrow big(x))$
The dog is big.	$\exists x, (dog(x) \wedge \forall y, (dog(y) \Rightarrow x = y) \wedge big(x))$

'The dog is not big' can be translated two ways:

It is not the case that the dog is big. (wide)	$\neg\exists x, (dog(x) \wedge \forall y, (dog(y) \Rightarrow x = y) \wedge big(x))$
The dog is non-big. (narrow)	$\exists x, (dog(x) \wedge \forall y, (dog(y) \Rightarrow x = y) \wedge \neg big(x))$

Using identity to symbolize quantities

There are at least _____ cookies.

This section writes variables with subscripts, so we can easily count them. In the text, QL contains only the variables a through z, but this was just to make them easy to write and type. In mathematics, you can use any variable name you want, including with subscripts. In *Lurch*, you can use long variable names, such as *gertrude*, even though QL officially restricts you to single letters.

one $\exists x, coo(x)$
two $\exists x_1, \exists x_2, (coo(x_1) \wedge coo(x_2) \wedge x_1 \neq x_2)$
three $\exists x_1, \exists x_2, \exists x_3, (coo(x_1) \wedge coo(x_2) \wedge coo(x_3) \wedge x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_2 \neq x_3)$
four $\exists x_1, \exists x_2, \exists x_3, \exists x_4, (coo(x_1) \wedge coo(x_2) \wedge coo(x_3) \wedge coo(x_4) \wedge x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_1 \neq x_4 \wedge x_2 \neq x_3 \wedge x_2 \neq x_4 \wedge x_3 \neq x_4)$
n $\exists x_1, \cdots, \exists x_n, (coo(x_1) \wedge \cdots \wedge coo(x_n) \wedge x_1 \neq x_2 \wedge \cdots \wedge x_{n-1} \neq x_n)$

There are at most _____ cookies.

One way to say 'at most n things are cookies' is to put a negation sign in front of one of the symbolizations above and say $\neg$'at least $n+1$ things are cookies.' Equivalently:

one $\forall x_1, \forall x_2, \big((coo(x_1) \wedge coo(x_2)) \Rightarrow x_1 = x_2\big)$
two $\forall x_1, \forall x_2, \forall x_3, \big((coo(x_1) \wedge coo(x_2) \wedge coo(x_3)) \Rightarrow (x_1 = x_2 \vee x_1 = x_3 \vee x_2 = x_3)\big)$
three $\forall x_1, \forall x_2, \forall x_3, \forall x_4, \big((coo(x_1) \wedge coo(x_2) \wedge coo(x_3) \wedge coo(x_4)) \Rightarrow (x_1 = x_2 \vee x_1 = x_3 \vee x_1 = x_4 \vee x_2 = x_3 \vee x_2 = x_4 \vee x_3 = x_4)\big)$
n $\forall x_1, \cdots, \forall x_{n+1}, \big((coo(x_1) \wedge \cdots \wedge coo(x_{n+1})) \Rightarrow (x_1 = x_2 \vee \cdots \vee x_n = x_{n+1})\big)$

There are exactly _____ cookies.

One way to say 'exactly n things are cookies' is to conjoin two of the symbolizations above and say 'at least n things are cookies' $\wedge$ 'at most n things are cookies.' The following equivalent formulae are shorter:

zero $\forall x, \neg coo(x)$
one $\exists x, \big(coo(x) \wedge \neg \exists y, (coo(y) \wedge x \neq y)\big)$
two $\exists x_1, \exists x_2, \big(coo(x_1) \wedge coo(x_2) \wedge x_1 \neq x_2 \wedge \neg \exists y, \big(coo(y) \wedge y \neq x_1 \wedge y \neq x_2\big)\big)$
three $\exists x_1, \exists x_2, \exists x_3, \big(coo(x_1) \wedge coo(x_2) \wedge coo(x_3) \wedge x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_2 \neq x_3 \wedge \neg \exists y, (coo(y) \wedge y \neq x_1 \wedge y \neq x_2 \wedge y \neq x_3)\big)$
n $\exists x_1, \cdots, \exists x_n, \big(coo(x_1) \wedge \cdots \wedge coo(x_n) \wedge x_1 \neq x_2 \wedge \cdots \wedge x_{n-1} \neq x_n \wedge \neg \exists y, (coo(y) \wedge y \neq x_1 \wedge \cdots \wedge y \neq x_n)\big)$

Specifying the size of the UD

Removing *coo* from the symbolizations above produces sentences that talk about the size of the UD. For instance, 'there are at least 2 things (in the UD)' may be symbolized as $\exists x, \exists y, x \neq y$.

Sometimes it is easier to show something by providing proofs than it is by providing models. Sometimes it is the other way round.

	YES	NO
Is $\mathcal{A}$ a tautology?	prove $\vdash \mathcal{A}$	give a model in which $\mathcal{A}$ is false
Is $\mathcal{A}$ a contradiction?	prove $\vdash \neg\mathcal{A}$	give a model in which $\mathcal{A}$ is true
Is $\mathcal{A}$ contingent?	give a model in which $\mathcal{A}$ is true and another in which $\mathcal{A}$ is false	prove $\vdash \mathcal{A}$ or $\vdash \neg\mathcal{A}$
Are $\mathcal{A}$ and $\mathcal{B}$ equivalent?	prove $\mathcal{A} \vdash \mathcal{B}$ and $\mathcal{B} \vdash \mathcal{A}$	give a model in which $\mathcal{A}$ and $\mathcal{B}$ 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 $\mathcal{B}$ and $\neg\mathcal{B}$
Is the argument '$\mathcal{P}$, $\therefore$ $\mathcal{C}$' valid?	prove $\mathcal{P} \vdash \mathcal{C}$	give a model in which $\mathcal{P}$ is true and $\mathcal{C}$ is false

Basic Rules of Proof

Reiteration

$m.$	$\mathcal{A}$	
	$\mathcal{A}$	R $m.$

Conjunction Introduction

$m.$	$\mathcal{A}$	
$n.$	$\mathcal{B}$	
	$\mathcal{A} \wedge \mathcal{B}$	$\wedge$I $m., n.$

Conjunction Elimination

$m.$	$\mathcal{A} \wedge \mathcal{B}$	
	$\mathcal{A}$	$\wedge$E $m.$
	$\mathcal{B}$	$\wedge$E $m.$

Disjunction Introduction

$m.$	$\mathcal{A}$	
	$\mathcal{A} \vee \mathcal{B}$	$\vee$I $m.$
	$\mathcal{B} \vee \mathcal{A}$	$\vee$I $m.$

Disjunction Elimination

$m.$	$\mathcal{A} \vee \mathcal{B}$	
$n.$	$\neg\mathcal{B}$	
	$\mathcal{A}$	$\vee$E $m., n.$

$m.$	$\mathcal{A} \vee \mathcal{B}$	
$n.$	$\neg\mathcal{A}$	
	$\mathcal{B}$	$\vee$E $m., n.$

Conditional Introduction

$m.$		$\mathcal{A}$	want $\mathcal{B}$
$n.$		$\mathcal{B}$	
	$\mathcal{A} \Rightarrow \mathcal{B}$		$\Rightarrow$I $m., n.$

Conditional Elimination

$m.$	$\mathcal{A} \Rightarrow \mathcal{B}$	
$n.$	$\mathcal{A}$	
	$\mathcal{B}$	$\Rightarrow$E $m., n.$

Biconditional Introduction

$m.$	$\mathcal{A} \Rightarrow \mathcal{B}$	
$n.$	$\mathcal{B} \Rightarrow \mathcal{A}$	
	$\mathcal{A} \Leftrightarrow \mathcal{B}$	$\Leftrightarrow$I $m., n.$

Biconditional Elimination

$m.$	$\mathcal{A} \Leftrightarrow \mathcal{B}$	
$n.$	$\mathcal{A}$	
	$\mathcal{B}$	$\Leftrightarrow$E $m., n.$

$m.$	$\mathcal{A} \Leftrightarrow \mathcal{B}$	
$n.$	$\mathcal{B}$	
	$\mathcal{A}$	$\Leftrightarrow$E $m., n.$

Negation Introduction

$m.$		$\mathcal{A}$	for reductio
$n.$		$\mathcal{B}$	
$p.$		$\neg\mathcal{B}$	
	$\neg\mathcal{A}$		$\neg$I $m., n., p.$

Negation Elimination

$m.$		$\neg\mathcal{A}$	for reductio
$n.$		$\mathcal{B}$	
$p.$		$\neg\mathcal{B}$	
	$\mathcal{A}$		$\neg$E $m., n., p.$

Quantifier Rules

Existential Introduction

$m.$	$\mathcal{A}[\chi = t]$	
	$\exists \chi, \mathcal{A}$	$\exists$I $m.$

Existential Elimination

$m.$	$\exists \chi, \mathcal{A}$	
$n.$	Let C be such that:	
	$\mathcal{A}[\chi = \mathcal{C}]$	$\exists$E $m.$

A constant can only be declared like this if it has not appeared in the proof yet. Line $n.$ must appear immediately before the conclusion, so that together they form a complete sentence.

Universal Introduction

$m.$	Let a be arbitrary.	
$n.$	$\mathcal{A}$	
	$\forall \chi, \mathcal{A}[a = \chi]$	$\forall$I $m., n.$

A constant can only be declared like this if it has not appeared in the proof yet, or only appeared in a subproof now closed.

Universal Elimination

$m.$	$\forall \chi, \mathcal{A}$	
	$\mathcal{A}[\chi = t]$	$\forall$E $m.$

Identity Rules

$t = t$	=I

$m.$	$a = b$	
$n.$	$\mathcal{A}$	
	$\mathcal{A}[a \sim b]$	=E $m., n.$
	$\mathcal{A}[b \sim a]$	=E $m., n.$

One constant may replace some or all occurrences of the other.

Derived Rules

Dilemma

$m.$	$\mathcal{A} \vee \mathcal{B}$	
$n.$	$\mathcal{A} \Rightarrow \mathcal{C}$	
$p.$	$\mathcal{B} \Rightarrow \mathcal{C}$	
	$\mathcal{C}$	$\vee *$ $m., n., p.$

Modus Tollens

$m.$	$\mathcal{A} \Rightarrow \mathcal{B}$	
$n.$	$\neg \mathcal{B}$	
	$\neg \mathcal{A}$	MT $m., n.$

Hypothetical Syllogism

$m.$	$\mathcal{A} \Rightarrow \mathcal{B}$	
$n.$	$\mathcal{B} \Rightarrow \mathcal{C}$	
	$\mathcal{A} \Rightarrow \mathcal{C}$	HS $m., n.$

Logical Equivalences

Commutivity (Comm)
$$(\mathcal{A} \wedge \mathcal{B}) \iff (\mathcal{B} \wedge \mathcal{A})$$
$$(\mathcal{A} \vee \mathcal{B}) \iff (\mathcal{B} \vee \mathcal{A})$$
$$(\mathcal{A} \Leftrightarrow \mathcal{B}) \iff (\mathcal{B} \Leftrightarrow \mathcal{A})$$

DeMorgan (DeM)
$$\neg(\mathcal{A} \vee \mathcal{B}) \iff (\neg \mathcal{A} \wedge \neg \mathcal{B})$$
$$\neg(\mathcal{A} \wedge \mathcal{B}) \iff (\neg \mathcal{A} \vee \neg \mathcal{B})$$

Double Negation (DN)
$$\neg\neg \mathcal{A} \iff \mathcal{A}$$

Material Conditional (MC)
$$(\mathcal{A} \Rightarrow \mathcal{B}) \iff (\neg \mathcal{A} \vee \mathcal{B})$$
$$(\mathcal{A} \vee \mathcal{B}) \iff (\neg \mathcal{A} \Rightarrow \mathcal{B})$$

Biconditional Exchange ($\Leftrightarrow$ex)
$$((\mathcal{A} \Rightarrow \mathcal{B}) \wedge (\mathcal{B} \Rightarrow \mathcal{A})) \iff (\mathcal{A} \Leftrightarrow \mathcal{B})$$

Quantifier Negation (QN)
$$\neg \forall \chi, \mathcal{A} \iff \exists \chi, \neg \mathcal{A}$$
$$\neg \exists \chi, \mathcal{A} \iff \forall \chi, \neg \mathcal{A}$$

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.

Nathan Carter is an associate professor of mathematical sciences in Waltham, Massachusetts. He uses computers to do mathematics, and to help people learn to do mathematics.